Second Generation Model-based Testing

Provably Strong Testing Methods for the Certification of Autonomous Systems
Part II of III –
Provably Strong Testing Methods for Autonomous Systems

Jan Peleska
University of Bremen and Verified Systems International GmbH
peleska@uni-bremen.de
2019-03-21
A Development Approach – the Basis for Model-based Testing
Typical architecture of an autonomous system

1. **Strategic layer**
   - Mission and Strategy
   - Mission state analysis
   - Mission planning
   - Rules

2. **Tactical layer**
   - Situation analysis
   - Situation Awareness
   - Task planning
   - Tasks

3. **Control layer**
   - Sensor data analysis
   - Actuator control
   - Commands

4. **Physical layer**
   - Sensors
   - Environment
   - Actuators

Identify applicable scenario from finite library of pre-defined parametrised scenarios
Mission and Strategy \rightarrow Mission state analysis \rightarrow Mission planning

Situation analysis \rightarrow Situation Awareness

Sensor data analysis

Sensors \rightarrow Environment

Actuators

Rules

Tasks

Mission

Strategic layer

Tactical layer

Control layer

Physical layer

Scheduling of risk mitigation actions and mission accomplishment

Safety objective. Select optimal behavioural strategy that keeps risks at acceptable level, while optimising the mission reachability, as long as safety permits.
**Scene.** Snapshot of traffic and environment constellations

**Situation.** Scene experienced from the perspective of one traffic participant – the SUT

**Scenario.** A transition system whose computations are physically consistent sequences of situations

Events/actions trigger transitions between situations – either increasing or lowering the risk

---

Design Restrictions

- To ensure constant worst-case execution time boundaries...
  
  - … only a **bounded number of scenarios** is admissible (no synthesis of new scenarios during runtime)

  - … only a **bounded number of risk mitigation strategies** are admissible (no learning of new mitigation strategies during runtime)
Design Workflow and MBT-Test Preparation

Scenario Identification


For each scenario, ...

Numerous publications, e.g.


Important research direction for autonomous systems
Runtime hazard identification instead of handling pre-specified hazards only
For each scenario, …
For each scenario, ...

Risk structure is created on abstraction

**Risk State Space:** hazard-related predicates $p_{H_1}, \ldots, p_{H_m}$

**Abstract State Space:** predicates $p_1(v_1, \ldots, v_n), \ldots, p_k(v_{k1}, \ldots)$

**CPS State Space:** variables $v_1, \ldots, v_n$

**Physical World**
For each scenario, ...

Scenario Identification → Hazard Analysis → Hazard Mitigation Strategy → Safety Monitor – Behavioural Model

Finite State Machine or SysML State Machine or Kripke Structure or CSP model or RoboChart or ...

Risk Structure
For each scenario, ...

Safety Monitor triggers mitigation actions for risk minimisation
Example. Creating a CSP Model for a Scenario-specific Safety Monitor
1. **Scenario.** Red car overtakes ego vehicle (blue car) and swerves into right lane.

2. 

3. 

Blue: Ego Vehicle
Variables if the CPS state space (scenario-independent)

Sensor data and actuator data (no further details shown)

\( t \)  
Time

\( \vec{x}_{\text{blue}} \)  
Position of blue car

\( \vec{x}_{\text{red}} \)  
Position of red car

\( \vec{v}_{\text{blue}} \)  
Speed of blue car

\( \vec{v}_{\text{red}} \)  
Speed of red car

\( \vec{a}_{\text{blue}} \)  
Acceleration of blue car

\( \vec{a}_{\text{red}} \)  
Acceleration of red car
Variables in the abstract state space ("predicate space")

\(d_{-2}, d_{-1}, d_0, d_1, d_2\)  
Relative distance thresholds red car/blue car
-2 : “red car is far behind blue car”,
-1 : “close behind”
0 : “next to”
1 : “close in front”
2 : “far in front”

\[ d_{-2} \equiv \| \vec{x}_{blue} - \vec{x}_{red} \| > \delta_{far} \land pr_1(\vec{x}_{blue}) - pr_1(\vec{x}_{red}) > 0 \]

\[ \ldots \]

\[ d_0 \equiv \| \vec{x}_{blue} - \vec{x}_{red} \| < \varepsilon \]

\[ \ldots \]

\[ d_2 \equiv \| \vec{x}_{blue} - \vec{x}_{red} \| > \delta_{far} \land pr_1(\vec{x}_{blue}) - pr_1(\vec{x}_{red}) < 0 \]
Variables in the abstract state space ("predicate space")

\( v_-, v_0, v_+ \)

Relative speed thresholds red car/blue car
- : “red car is much slower than blue car”
0 : “red and blue car have the same speed”
1 : “red car is faster than blue car”

\[ v_- \equiv \| \vec{v}_{blue} - \vec{v}_{red} \| > \sigma \land pr_1(\vec{v}_{blue} - \vec{v}_{red}) > 0 \]

...
Variables in the abstract state space ("predicate space")

\[ \ell_{blue}, \ell_{red}, r_{blue}, r_{red}, s_{blue}, s_{red} \]

Blue car and red car, respectively, are in left lane / right lane / continue straight

\[ r_{red} \equiv pr_2(\vec{x}_{red}) < mid \]

...

\[ R_{blue}, L_{blue}, R_{red}, L_{red} \]

Blue car and red car change to the right lane or in the left lane, respectively

\[ R_{red} \equiv pr_2(\frac{\vec{v}_{red}}{||\vec{v}_{red}||}) < -\gamma < 0 \]

...
Variables in the abstract state space ("predicate space")

\[ a_{-2}, a_{-1}, a_0, a_1, a_2 \]

Ego vehicle (blue car) accelerates in driving direction
-2: maximal brake force (negative acceleration)
-1: normal brake force
0: no acceleration
1: normal acceleration
2: maximal acceleration

\[ a_{-2} \equiv \| \vec{a}_\text{blue} \| \leq a_{\min} < 0 \]

...
Variables in the hazard space ("predicate space")

\[ h_1 \equiv \ell_{\text{red}} \land r_{\text{blue}} \land d_0 \land R_{\text{red}} \]

**Hazard \( h_1 \).**

The red car is in the left lane, the blue car is in the right lane, the cars are very close to each other, the **red car is swerving into the right lane**.
Result of hazard mitigation strategy: refined hazard

Mario Gleirscher, Stefan Kugele:
From Hazard Analysis to Hazard Mitigation Planning:

\[ h_{1.1} \equiv c_{red} \land r_{blue} \land d_0 \land R_{red} \land v_- \]

**Hazard \( h_{1.1} \).**
The red car is in the left lane,
the blue car is in the right lane,
the cars are very close to each other,
the red car is swerving into the right lane,
the red car is much slower than the blue car

**Admissible mitigation action.**
Maximal acceleration of blue car
Result of hazard mitigation strategy: refined hazard

\[ h_{1.2} \equiv \ell_{\text{red}} \land r_{\text{blue}} \land d_0 \land R_{\text{red}} \land v_0 \]

**Hazard h_{1.2}.**
The red car is in the left lane, the blue car is in the right lane, the cars are very close to each other, the **red car is swerving into the right lane**, the red car has same speed as the blue car

**Admissible mitigation actions.**
(1) Brake blue car with maximal force
(2) Maximal acceleration of blue car
Result of hazard mitigation strategy: refined hazard

\[ h_{1.3} \equiv \ell_{\text{red}} \land r_{\text{blue}} \land d_0 \land R_{\text{red}} \land v_+ \]

**Hazard h_{1.3}.**
The red car is in the left lane, the blue car is in the right lane, the cars are very close to each other, the red car is swerving into the right lane, the red car is faster than the blue car

**Admissible mitigation action.**
Brake blue car with maximal force
Derive Safety Monitor Model from Hazard Mitigation Analysis

Objectives for the safety monitor
1. Input predicates from the predicate state space
2. In hazard states, enforce hazard mitigation actions obtained from risk structure
3. Optimal mitigation actions force system into “acceptable risk corridor” and still allow for mission completion

Inputs to safety monitor – from predicate state space

\[ d_{-2}, d_{-1}, d_0, d_1, d_2 \]
\[ v_-, v_0, v_+ \]
\[ \ell_{\text{blue}}, \ell_{\text{red}}, r_{\text{blue}}, r_{\text{red}}, s_{\text{blue}}, s_{\text{red}} \]
\[ R_{\text{red}}, L_{\text{red}} \]

Outputs of safety monitor – from predicate state space

\[ R_{\text{blue}}, L_{\text{blue}} \]
\[ a_{-2}, a_{-1}, a_0, a_1, a_2 \]
Interplay Between Mission Planning and Safety Monitor

Predicate space data relevant for mission planning

Mission Planning

Safety Monitor

$R_{plan}^{plan}, L_{plan}^{plan}$

$a_{2}^{plan}, a_{1}^{plan}, a_{0}^{plan}, a_{1}^{plan}, a_{2}^{plan}$

$R_{blue}, L_{blue}$

$a_{2}, a_{1}, a_{0}, a_{1}, a_{2}$
Nondeterministic CSP Model

Scenario1 = MissionPlanning1
   [ | { R_blue_plan, L_blue_plan, a_minus2_plan,
        a_minus1_plan, a_0_plan, a_1_plan, a_2_plan } | ]
   SafetyMonitor1

MissionPlanning1 = ( |~| e:{R_blue_plan, L_blue_plan,a_minus2_plan,
        a_minus1_plan, a_0_plan, a_1_plan, a_2_plan} @ e -> MissionPlanning1)
SafetyMonitor1 = FAR(0)

FAR(vRel) = l_blue -> Scenario2
    [ ]
    ... [ ]
    r_red -> Scenario3
    [ ]
    ... [ ]
    d_minus1 -> NEAR(vRel)
    [ ]
    d_0 -> CLOSE(vRel)
    [ ]
    d_1 -> SafetyMonitor1
    [ ]
    d_2 -> SafetyMonitor1
    [ ]
    v_minus -> FAR(-1)
    [ ]
    v_0 -> FAR(0)
    [ ]
    v_plus -> FAR(1)
    [ ]
    L_blue_plan -> L_blue -> FAR(vRel)
    [ ]
    R_blue_plan -> FAR(vRel)
    [ ]
    a_minus2_plan -> a_minus1 -> FAR(vRel)
    [ ]
    a_minus1_plan -> a_minus1 -> FAR(vRel)
    ... [ ]
    a_2_plan -> a_1 -> FAR(vRel)
\[\text{NEAR}(v_{\text{Rel}}) = l_{\text{blue}} \rightarrow \text{Scenario2} \]

\[
\ ...
\]

\[
\ r_{\text{red}} \rightarrow \text{Scenario3}
\]

\[
\ ...
\]

\[
\ d_{\text{minus2}} \rightarrow \text{FAR}(v_{\text{Rel}})
\]

\[
\ d_{\text{minus1}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ d_{0} \rightarrow \text{CLOSE}(v_{\text{Rel}})
\]

\[
\ d_{1} \rightarrow \text{SafetyMonitor1}
\]

\[
\ d_{2} \rightarrow \text{SafetyMonitor1}
\]

\[
\ v_{\text{minus}} \rightarrow \text{NEAR}(-1)
\]

\[
\ v_{0} \rightarrow \text{NEAR}(0)
\]

\[
\ v_{\text{plus}} \rightarrow \text{NEAR}(1)
\]

\[
\ (v_{\text{Rel}} \geq 0) \& L_{\text{blue}\text{-plan}} \rightarrow L_{\text{blue}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ (v_{\text{Rel}} < 0) \& L_{\text{blue}\text{-plan}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ R_{\text{blue}\text{-plan}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ a_{\text{minus2}\text{-plan}} \rightarrow a_{\text{minus1}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ a_{\text{minus1}\text{-plan}} \rightarrow a_{\text{minus1}} \rightarrow \text{NEAR}(v_{\text{Rel}})
\]

\[
\ ...
\]
CLOSE(vRel) = \text{scenario2}\n\text{...}\n\text{...}\n(vRel == 0) & R\_red \to (a_2 \to \text{scenario3})
|\sim|
a\_minus_2 \to \text{scenario4}\n(vRel == -1) & R\_red \to a_2 \to \text{scenario3}\n(vRel == 1) & R\_red \to a\_minus_2 \to \text{scenario4}\nd\_minus2 \to \text{FAR(vRel)}\n\text{...}\nv\_minus \to \text{CLOSE(-1)}\nv\_0 \to \text{CLOSE(0)}\nv\_plus \to \text{CLOSE(1)}\n\text{L\_blue\_plan} \to \text{CLOSE(vRel)}\n\text{R\_blue\_plan} \to \text{CLOSE(vRel)}\na\_minus2\_plan \to a\_minus1 \to \text{CLOSE(vRel)}\na\_minus1\_plan \to a\_minus1 \to \text{CLOSE(vRel)}\na\_0\_plan \to a\_0 \to \text{CLOSE(vRel)}\na\_1\_plan \to a\_1 \to \text{CLOSE(vRel)}\na\_2\_plan \to a\_1 \to \text{CLOSE(vRel)}
Per-Scenario MBT
Per-Scenario MBT

• Test strategy options – complete strategies exist for each option

  • Show I/O-equivalence of SUT with safety monitor

  • Show that SUT is a refinement of safety monitor (allows for nondeterministic models and SUTs)

    • This is explained in the breakout session

• Show that SUT implements safety-related requirements correctly
Learning Without Impairing Safety
Now where does learning fit in?

- What we can handle and probably get certified along the lines described above

- Allow behavioural optimisations in mission planning, because safety monitor masks unsafe learning effects

- Allow behavioural optimisations in control layer only within the limits of abstract trajectory given by the safety controller

- Additional runtime monitoring can supervise this and enforce that the control layer data remains in these limits
Now where does learning fit in?

- What we cannot handle today and probably wouldn’t get certified
  - Learn new hazards at runtime
  - Learn new mitigation actions at runtime
Further Research Points
Statistical Testing

• For validation testing, scenarios need to be tested with a statistically significant number of different environment behaviours (“red car” in our example)

• Formal approaches to combined system testing & statistical testing
  • Based on Probabilistic Automata, Markov Automata, Stochastic Automata

Equivalence Class Testing

- **Recall.** Safety monitor operates on abstracted predicate space

- But concrete testing needs to stimulate SUT with concrete values making some of these predicates true, others false

- **Complete equivalence testing theory** gives answers about how to select concrete data samples from predicates

Continuous Certification
Approach to autonomous cyber-physical systems (ACPS) certification

- Virtual certification = certification in simulation environment
- Deployment after re-certification via software upload
Retrospective View on Test-related Challenges
Test Case Generation – Challenges

- **Too many test cases** required to create them manually

- **No complete reference model** available for MBT, so model-based test generation does not necessarily lead to all relevant test cases

- Test models need comprehensive **environment representation**

- Some validation tests may need to be designed/executed during runtime – **runtime acceptance testing**:
  - Validation depends on contracts between configuration of constituent systems
  - Validation depends on mission details specified for the actual task at hand
Test Oracles – Challenges

• For autonomous systems, test oracles need to cope with

1. Behaviour that is under-specified

2. Behaviour that is only acceptable if its risk level is acceptable

3. Behaviour that is not deterministic, but follows some (sometimes even unknown) probability distribution or probabilistic reference model
Test Oracles – Challenges

- **Example 1. Under-specified behaviour**

  - A robot arm handing a drinking cup to a disabled patient can solve this mission by infinitely many trajectories for the cup.

  - This type of problem has led to layered architectures in robotics control software.
  - **Strategic Layer** for defining and controlling the high-level mission (“lift cup from table to patient’s mouth”)
  - **Control layer** for executing concrete movements in space-time (“find trajectory for cup to reach patient’s mouth without collisions with any obstacles”)
Test Oracles – Challenges

• **Example 2. Behaviour that is only acceptable if its risk level is acceptable**

  • An autonomous car avoiding collision with another car during conflicting lane changes by accelerating during the lane change – instead of aborting the lane change

  • Test fails due to intolerable risk taken by autonomous car E
Test Oracles – Challenges

- Example 3. Behaviour that is not deterministic, but follows some probabilistic reference model
- A drone that chooses landing trajectories that are distributed around an optimal trajectory with acceptable variance
Test Oracles – Challenges

• Test oracles in validation tests for autonomous systems will become a combination of
  • conventional oracles for control systems and
  • statistical testing of hypotheses

• For the statistical testing, the number of test executions (for the same test case) needs to be much higher than for deterministic or nondeterministic systems without statistical distribution requirements
Test Oracles

• For autonomous safety-critical systems (as in our case study) test oracles have extended verdicts
  
  • (definitely) FAIL – violation of a non-probabilistic requirement – the mission objectives could not be achieved

  • FAIL due to unacceptable risk level – though the mission objectives could be achieved

  • PASS with acceptable risk level – the mission objectives could be achieved

  • (definitely) PASS – conformance to a non-probabilistic requirement

  • INCONCLUSIVE
Final Remark

• In Zen Buddhism, there is the notion of the great doubt
  • Question every experience assumed to be true so far – even the experience of enlightenment

• This great doubt seems to be most appropriate for investigating new challenging research fields with potentially hazardous consequences for our society
PLEASE ATTEND THE BREAKOUT SESSION ON COMPLETE CSP REFINEMENT TESTING LATER TODAY!