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

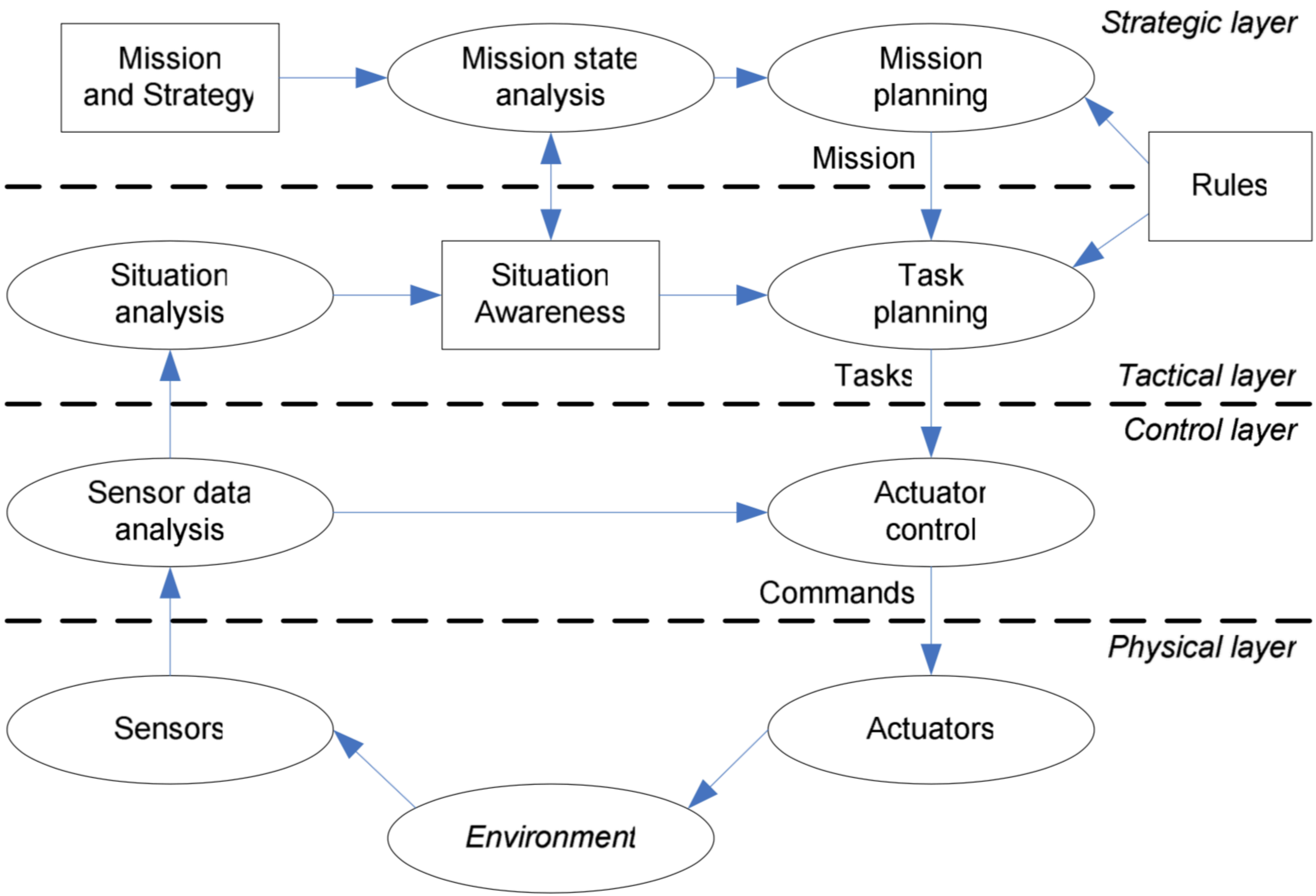
Jan Peleska
University of Bremen and Verified Systems International GmbH
peleska@uni-bremen.de
2019-03-21

Universität Bremen

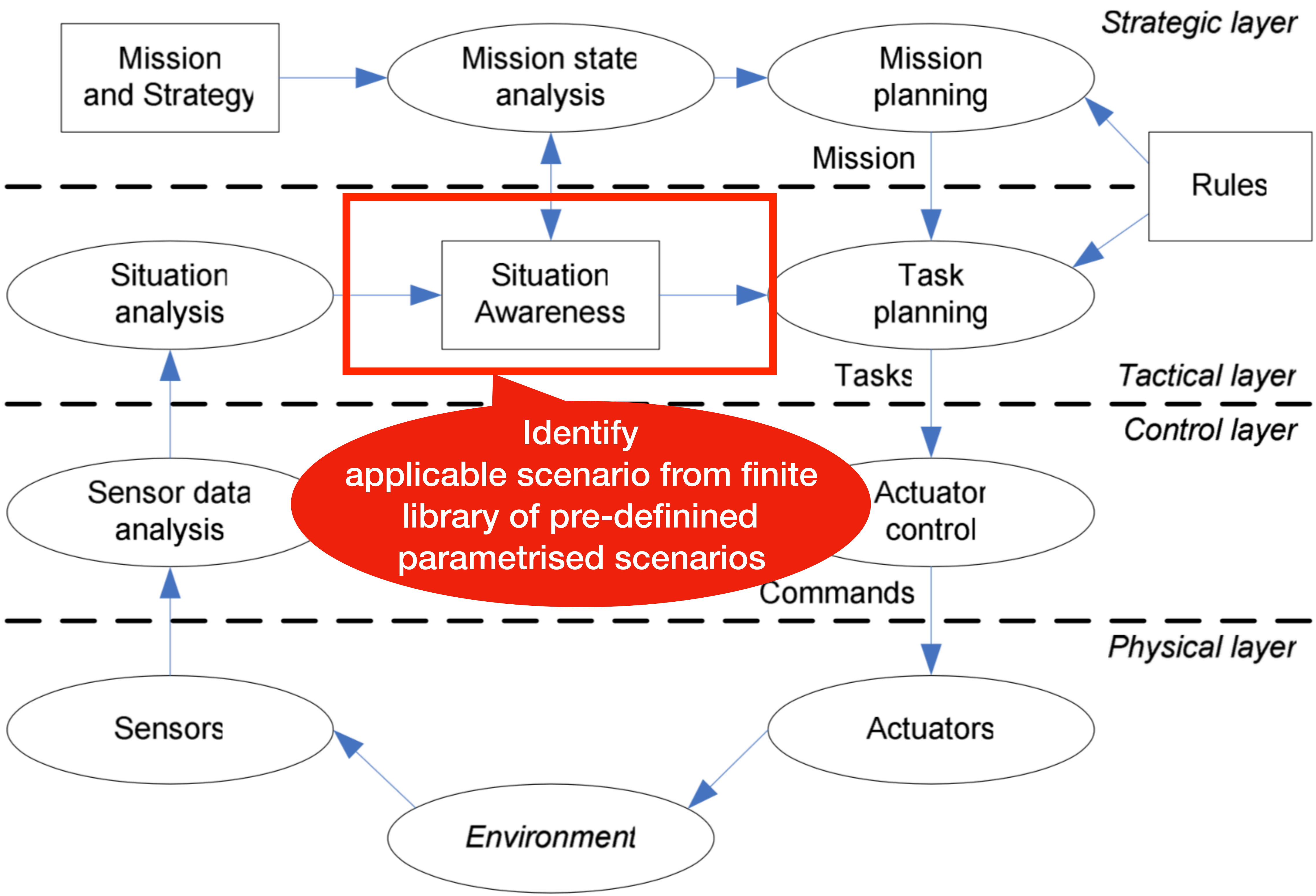
Verified

A Development Approach – the Basis for Model-based Testing

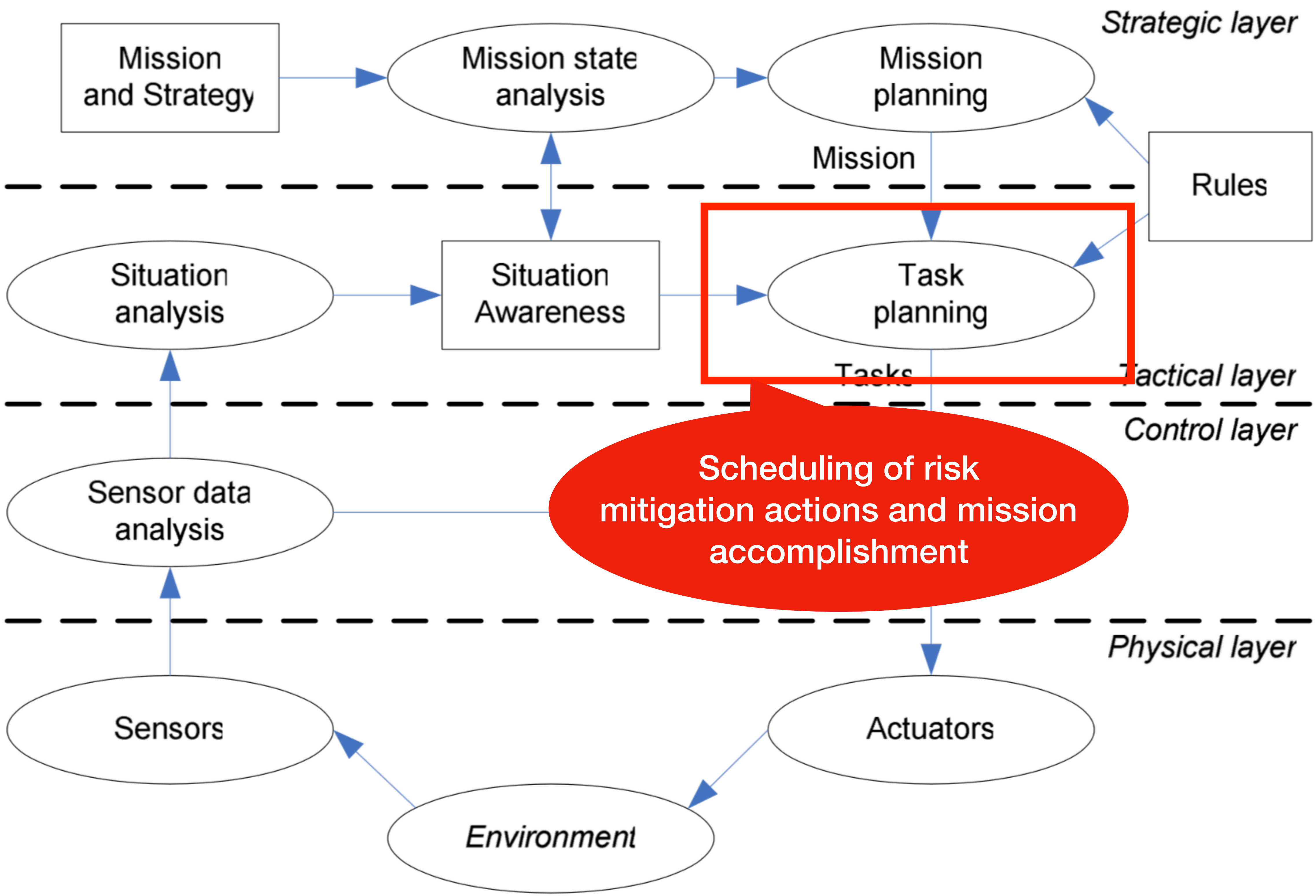
Typical architecture of an autonomous system



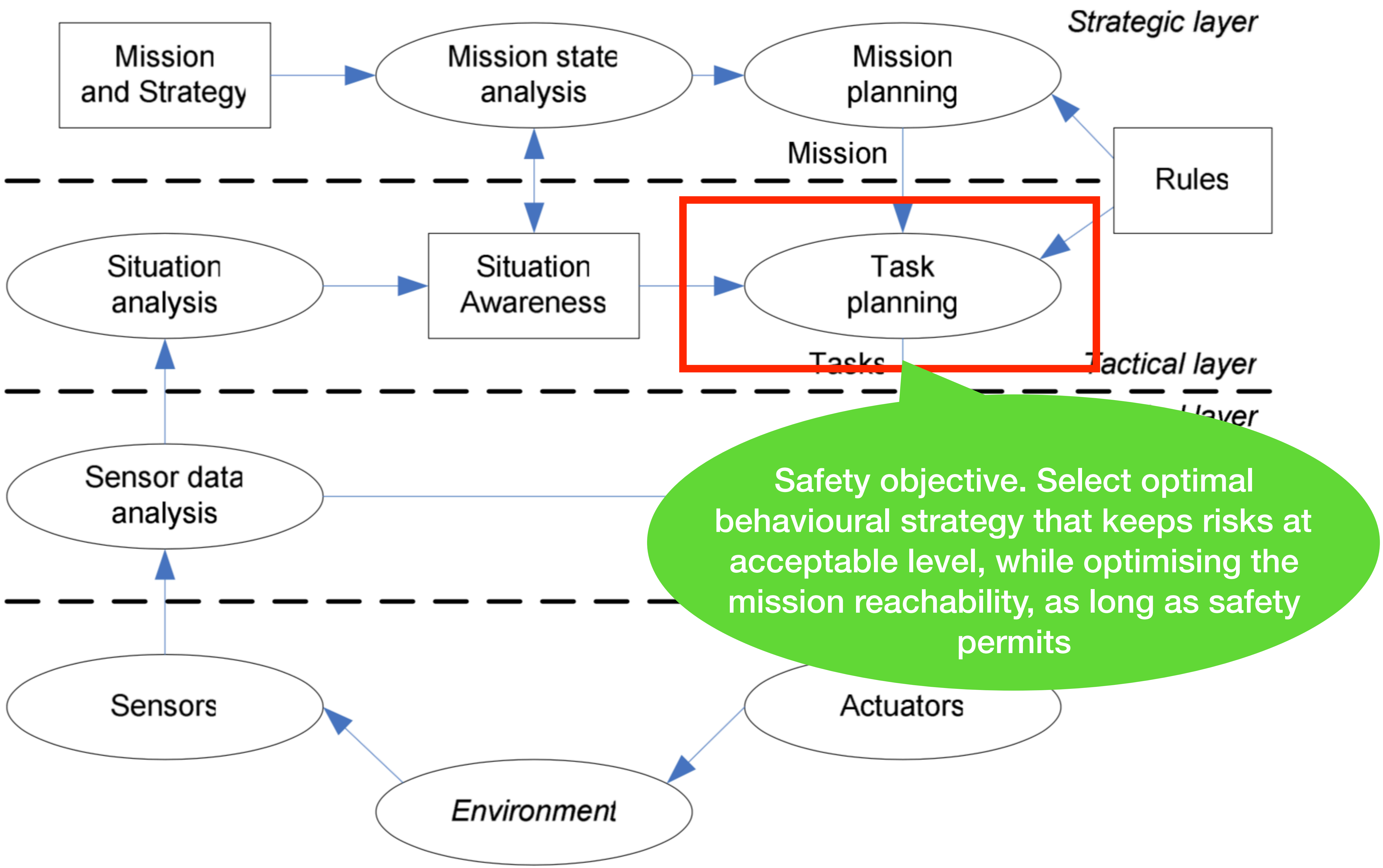
Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg



Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg



Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg



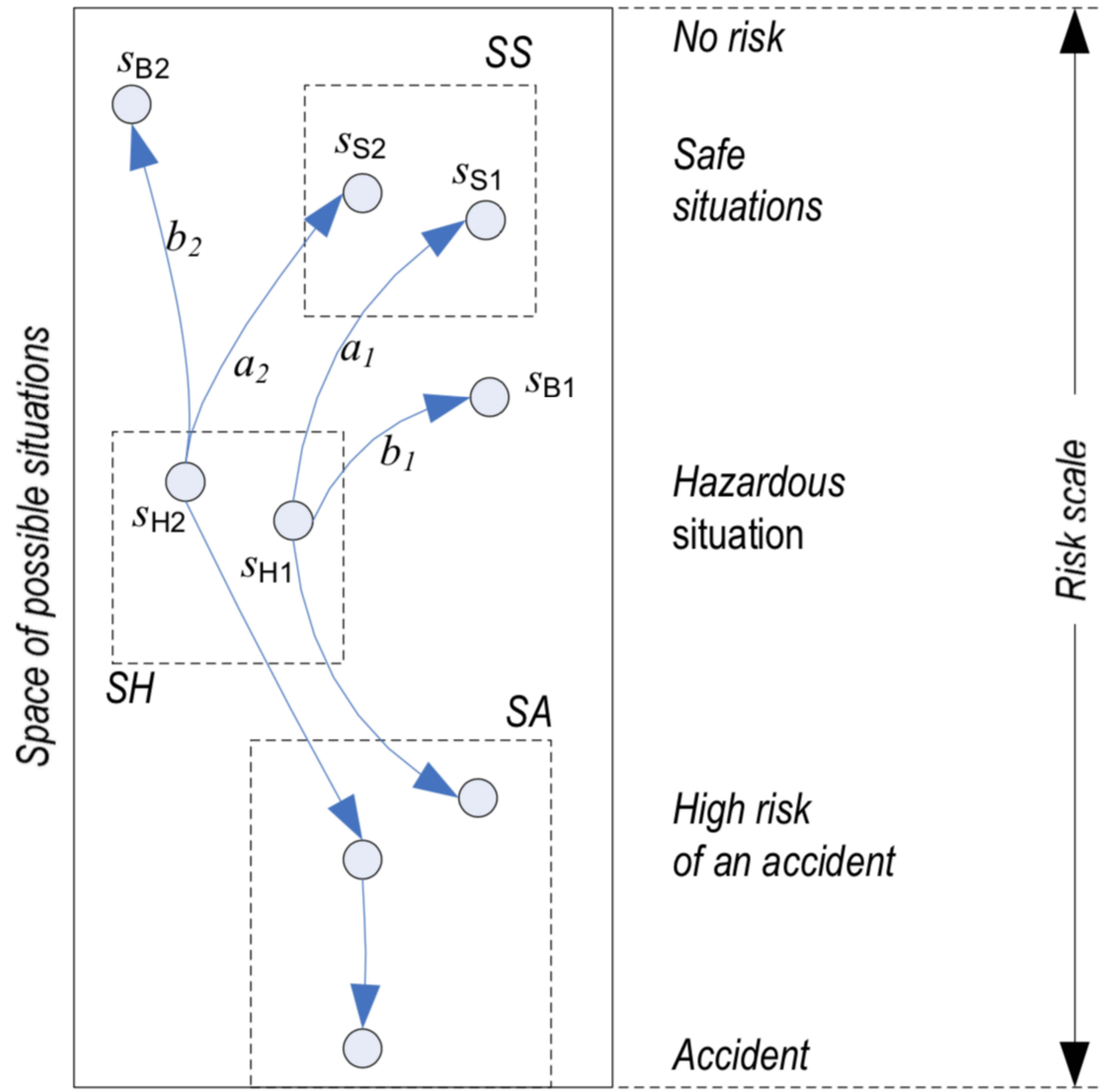
Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

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



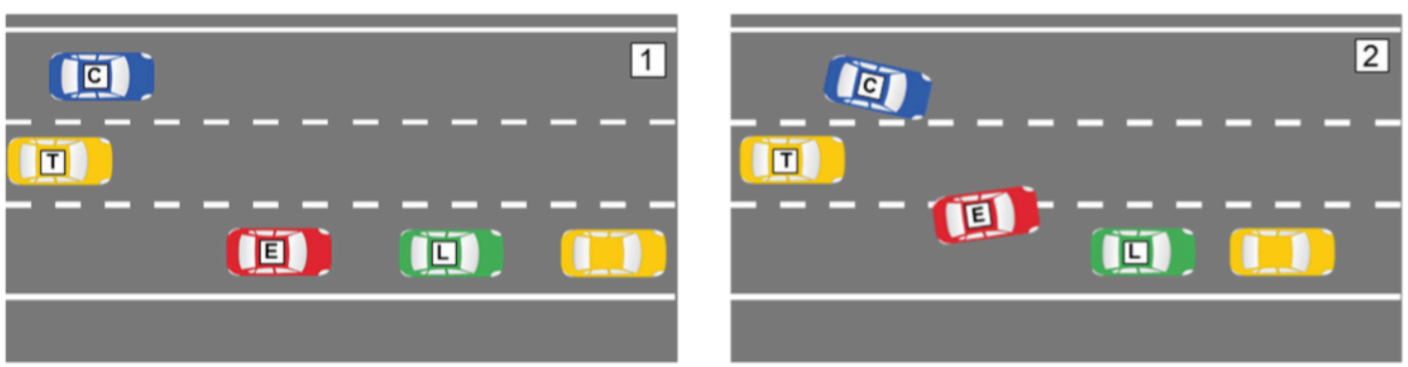
Wardziński A. (2008) Safety Assurance Strategies for Autonomous Vehicles. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

Design Restrictions

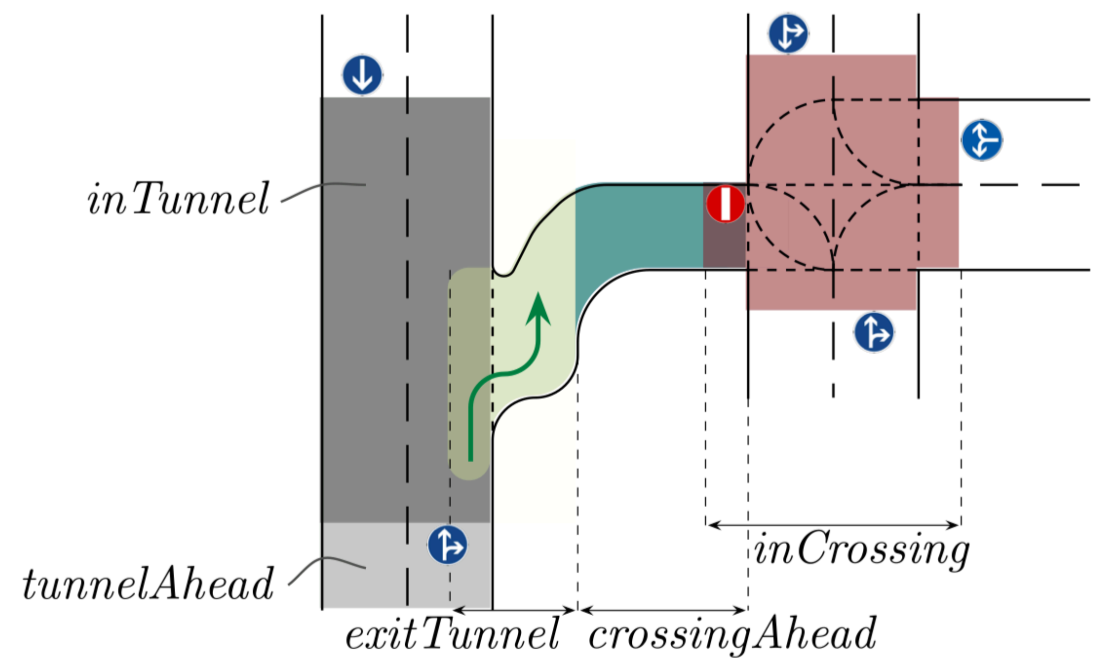
- To ensure constant worst-case execution time boundaries …
  - … only a **bounded number of scenarios** is admissible (no synthesis of new scenarios during runtime)
  - … only a **bounded number of risk mitigation strategies** are admissible (no learning of new mitigation strategies during runtime)

Design Workflow and MBT-Test Preparation

Scenario Identification



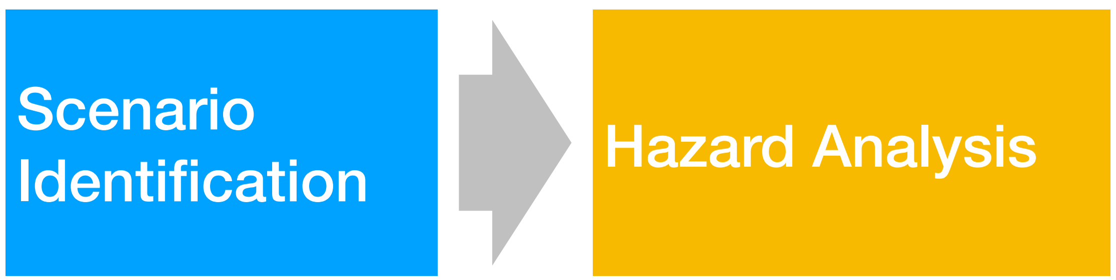
Hardi Hungar: Scenario-Based Validation of Automated Driving Systems. ISoLA (3) 2018: 449-460



Ulbrich, S., et al.: Defining and substantiating the terms scene, situation and scenario for automated driving. In: IEEE International Annual Conference on Intelligent Transportation Systems (ITSC) (2015)

Mario Gleirscher, Stefan Kugele: **From Hazard Analysis to Hazard Mitigation Planning: The Automated Driving Case.** CoRR abs/1802.08327 (2018)

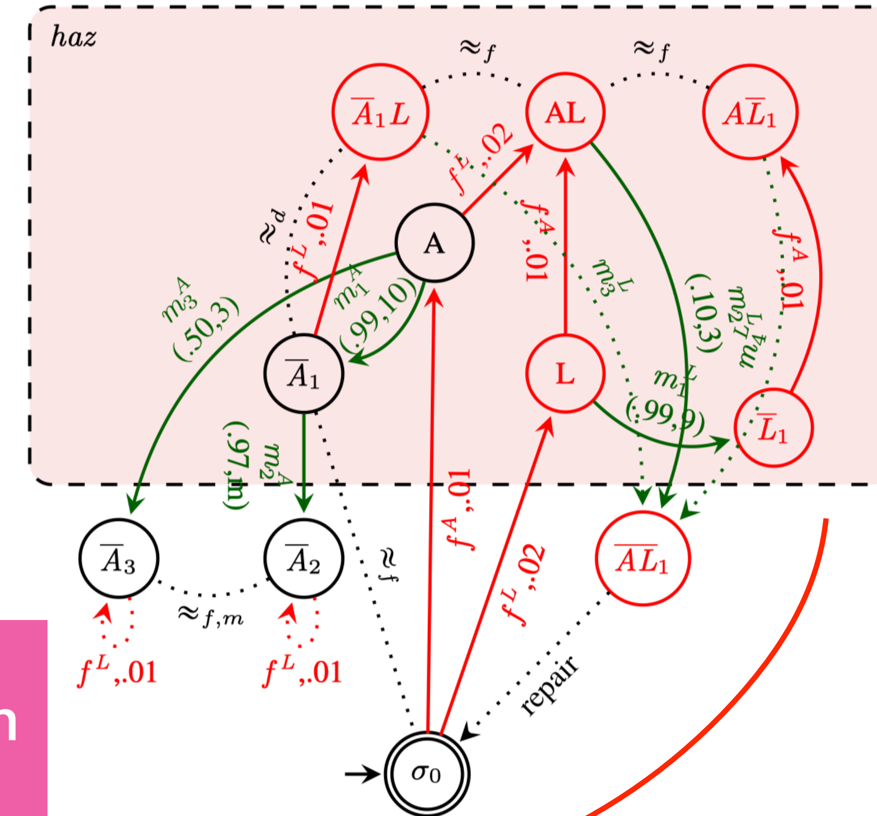
For each scenario, …



Numerous publications, e.g.

Mario Gleirscher:
Hazard Analysis for Technical Systems. SWQD 2013: 104-124

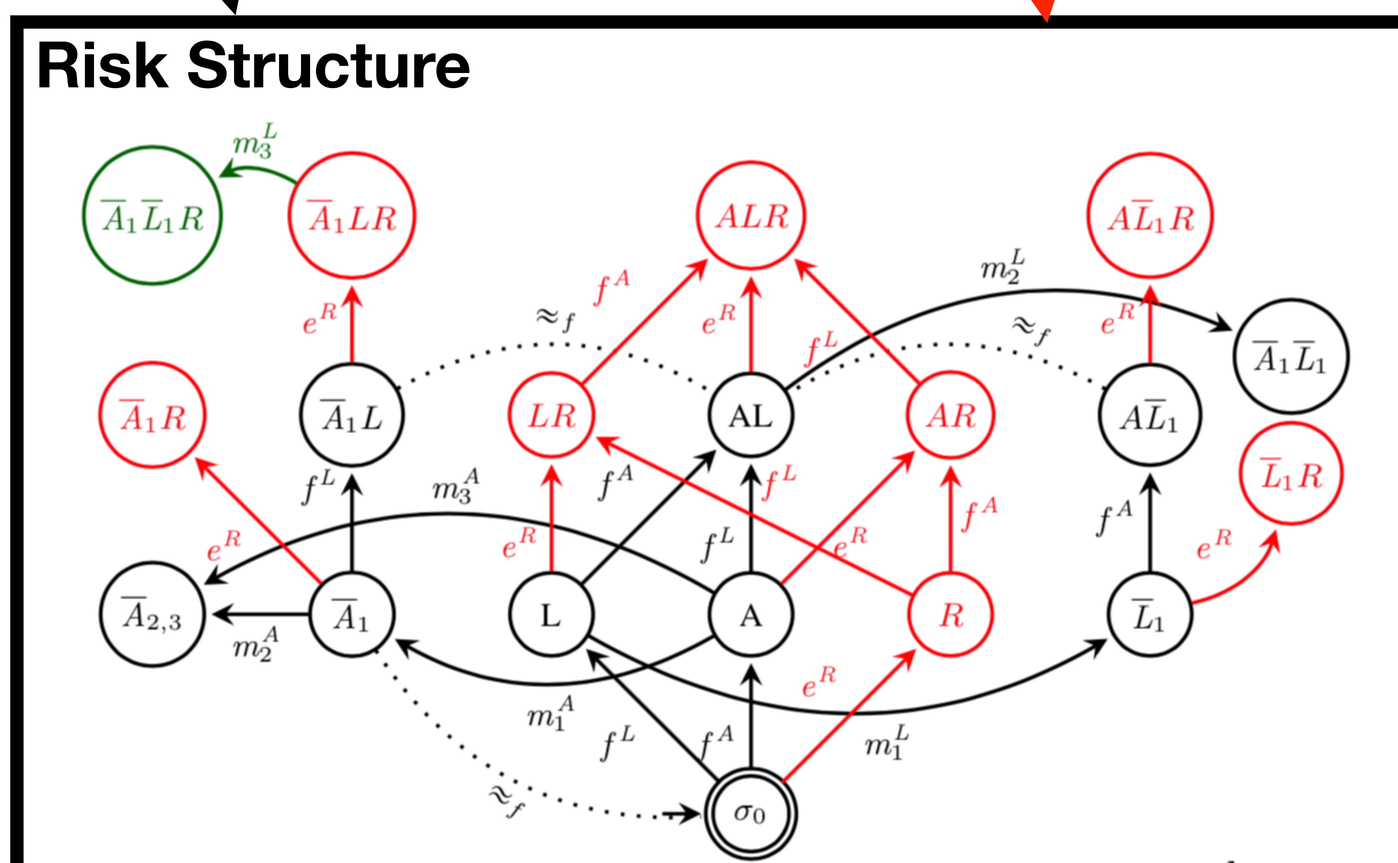
Important research direction for autonomous systems
Runtime hazard identification instead of handling pre-specified hazards only



For each scenario, …

Scenario Identification → Hazard Analysis → Hazard Mitigation Strategy

Incremental elaboration



Mario Gleirscher, Stefan Kugele:
From Hazard Analysis to Hazard Mitigation Planning: The Automated Driving Case. CoRR abs/1802.08327 (2018)

For each scenario, …

Scenario Identification → Hazard Analysis → Hazard Mitigation Strategy

Risk structure is created on abstraction

Risk State Space: hazard-related predicates

$p_{H_1}, \ldots, p_{H_m}$

↑

Abstract State Space: predicates

$p_1(v_1, \ldots, v_n), \ldots, p_k(v_{k_1}, \ldots)$

↑

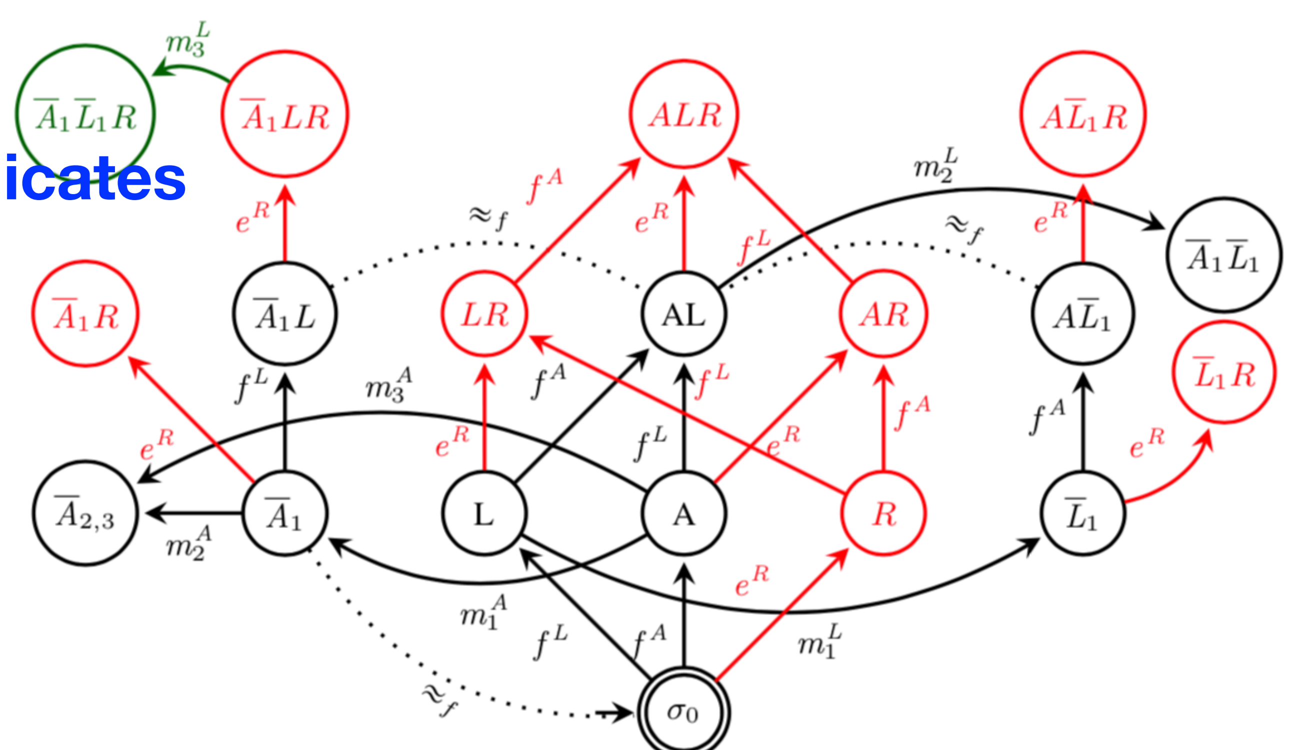
CPS State Space: variables

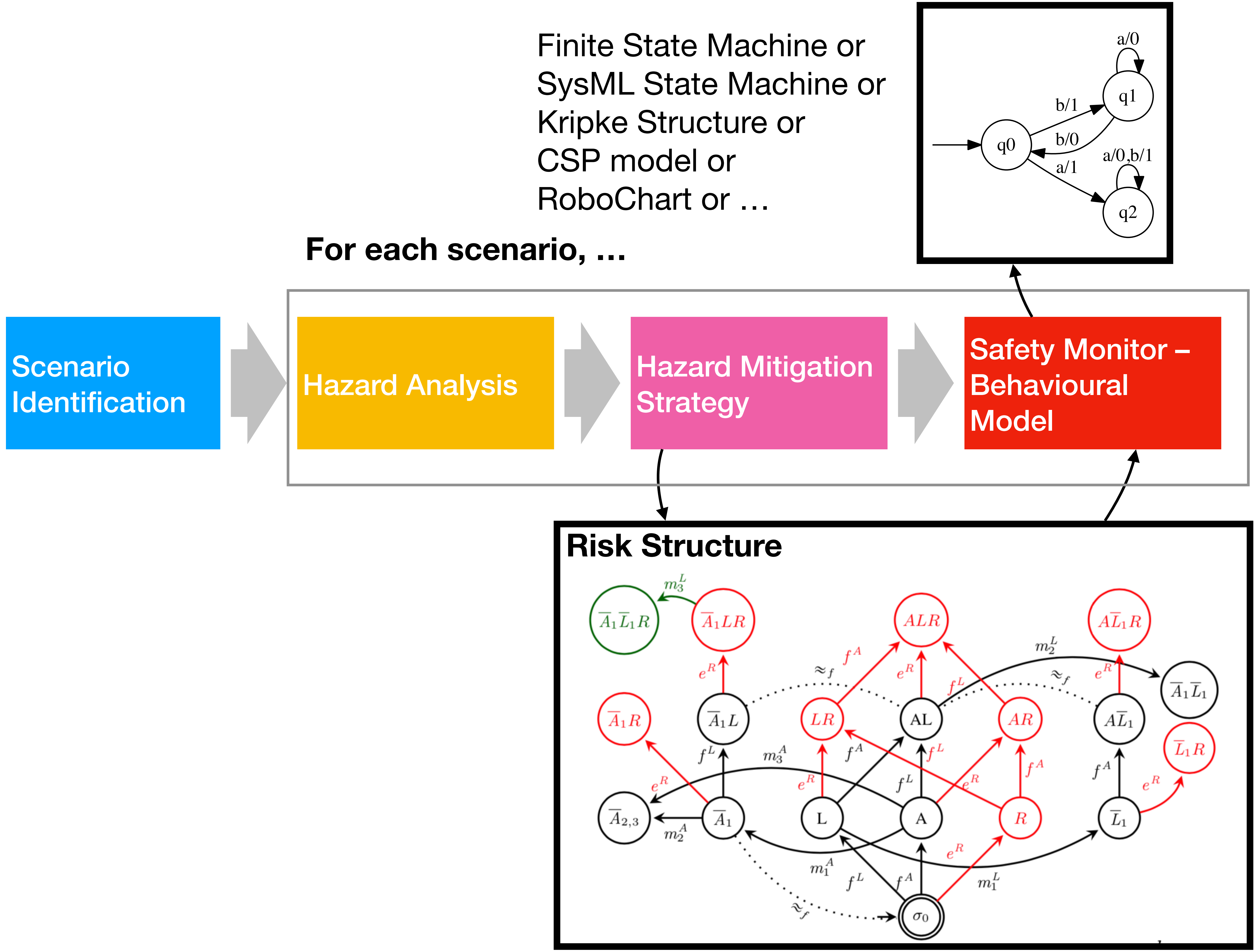
$v_1, \ldots, v_n$

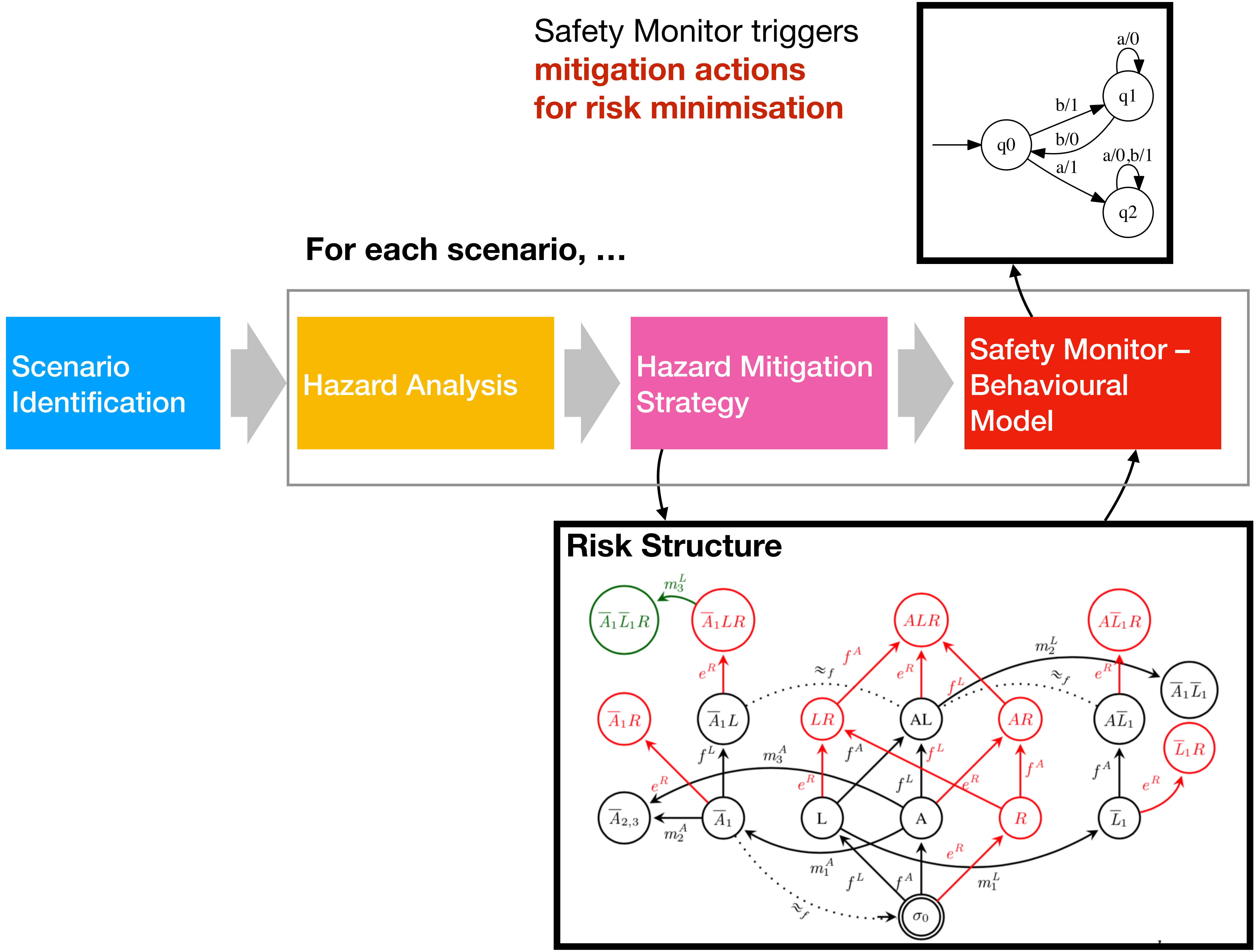
↑

Physical World

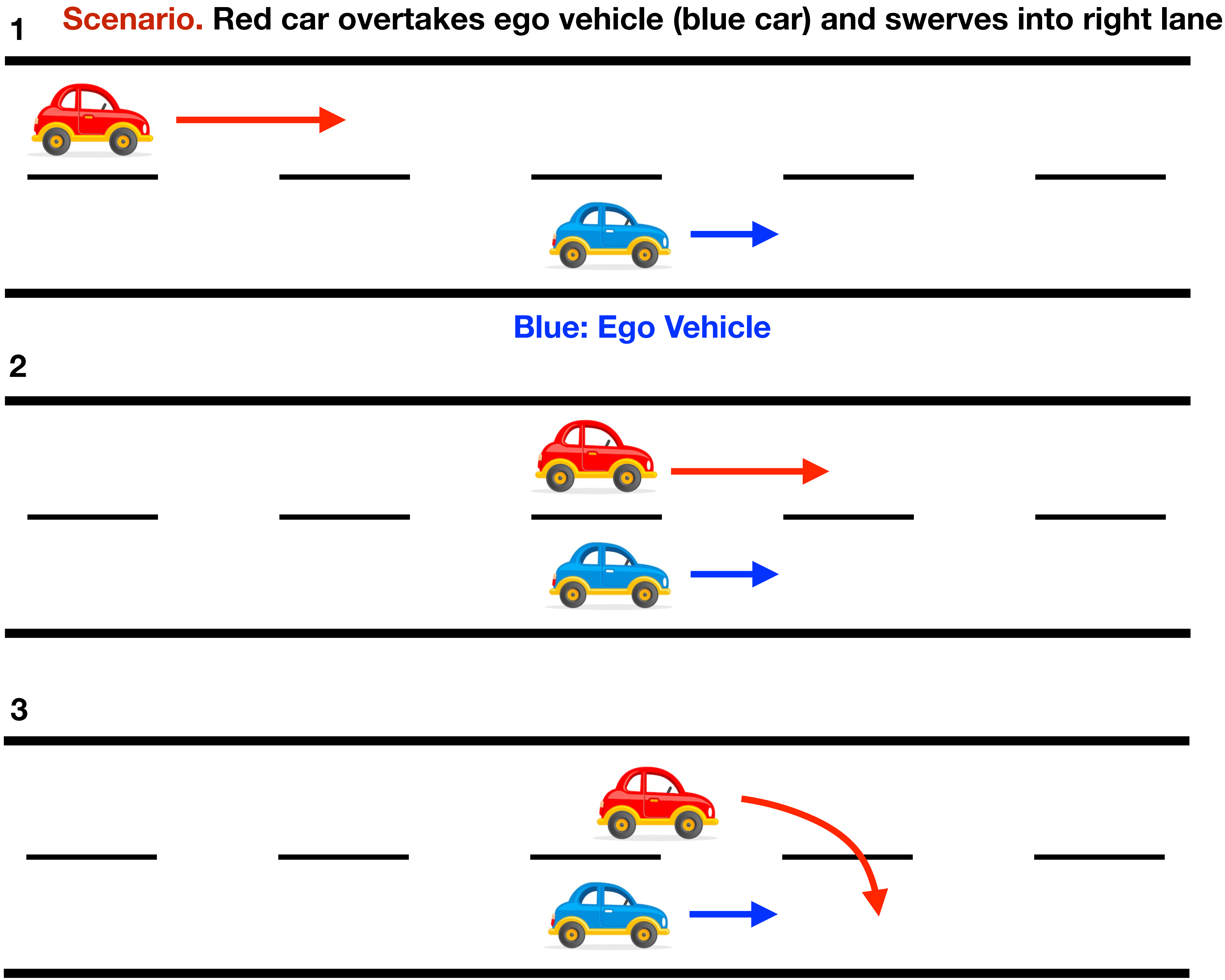
Risk Structure




Finite State Machine or
SysML State Machine or
Kripke Structure or
CSP model or
RoboChart or …
For each scenario, …
Scenario Identification
Hazard Analysis
Hazard Mitigation Strategy
Safety Monitor – Behavioural Model
Risk Structure
q0
q1
q2
a/0
b/1
b/0
a/1
a/0,b/1


Safety Monitor triggers
mitigation actions
for risk minimisation
For each scenario, ...
Scenario Identification
Hazard Analysis
Hazard Mitigation Strategy
Safety Monitor – Behavioural Model
Risk Structure
q0
q1
q2
a/0
b/1
b/0
a/1
a/0,b/1

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
Blue: Ego Vehicle
2
3

Variables if the CPS state space (scenario-independent)

	Sensor data and actuator data (no further details shown)
t	Time
$\overrightarrow{x}_{blue}$	Position of blue car
$\overrightarrow{x}_{red}$	Position of red car
$\overrightarrow{v}_{blue}$	Speed of blue car
$\overrightarrow{v}_{red}$	Speed of red car
$\overrightarrow{a}_{blue}$	Acceleration of blue car
$\overrightarrow{a}_{red}$	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"

$$d_{-2} \equiv \| \overrightarrow{x}_{blue} - \overrightarrow{x}_{red} \| > \delta_{far} \wedge pr_1(\overrightarrow{x}_{blue}) - pr_1(\overrightarrow{x}_{red}) > 0$$

...

$$d_0 \equiv \| \overrightarrow{x}_{blue} - \overrightarrow{x}_{red} \| < \varepsilon$$

...

$$d_2 \equiv \| \overrightarrow{x}_{blue} - \overrightarrow{x}_{red} \| > \delta_{far} \wedge pr_1(\overrightarrow{x}_{blue}) - pr_1(\overrightarrow{x}_{red}) < 0$$

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”

$$v_{-} \equiv \| \overrightarrow{v}_{blue} - \overrightarrow{v}_{red} \| > \sigma \wedge pr_1(\overrightarrow{v}_{blue} - \overrightarrow{v}_{red}) > 0$$

...

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

$$r_{red} \equiv pr_2(\overrightarrow{x}_{red}) < mid$$

...

$R_{blue}, L_{blue}, R_{red}, L_{red}$

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

$$R_{red} \equiv \; pr_2\left(\frac{\overrightarrow{v}_{red}}{\| \overrightarrow{v}_{red} \|}\right) < -\gamma < 0$$

...

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

$$a_{-2} \equiv \| \overrightarrow{a}_{blue} \| \leq a_{min} < 0$$

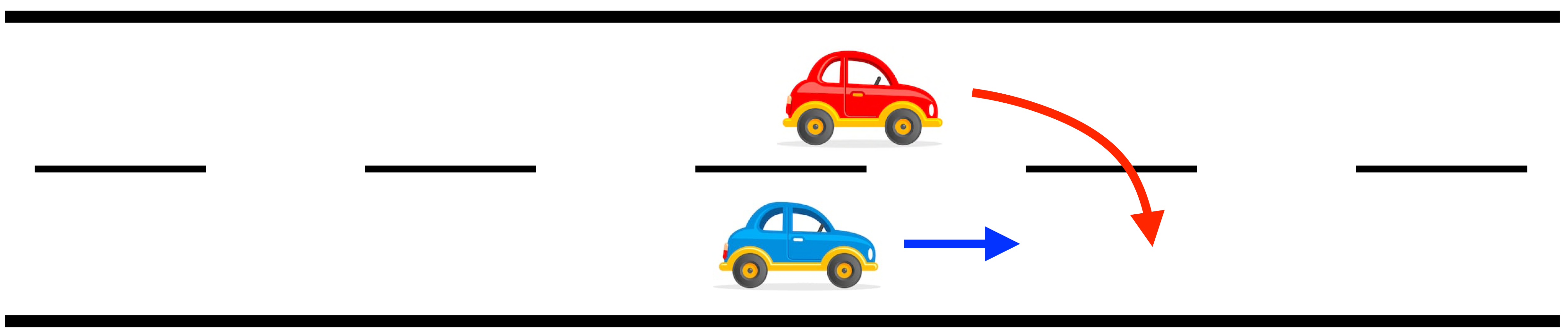
...

Variables in the hazard space ("predicate space")

$$h_1 \equiv \ell_{red} \wedge r_{blue} \wedge d_0 \wedge R_{red}$$

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**

3



Result of hazard mitigation strategy: refined hazard

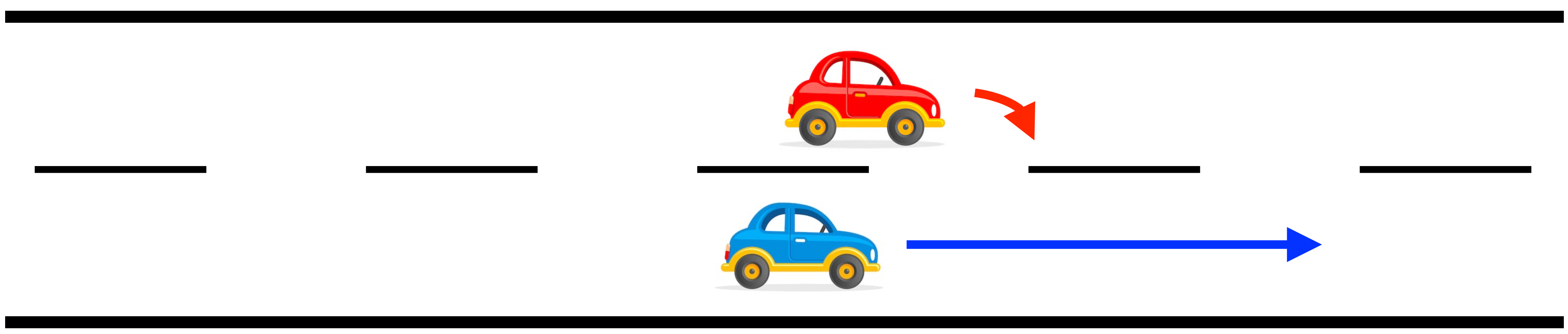
Mario Gleirscher, Stefan Kugele:
From Hazard Analysis to Hazard Mitigation Planning:
The Automated Driving Case. CoRR abs/1802.08327 (2018)

$$h_{1.1} \equiv \ell_{red} \wedge r_{blue} \wedge d_0 \wedge R_{red} \wedge v_-$$

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

3



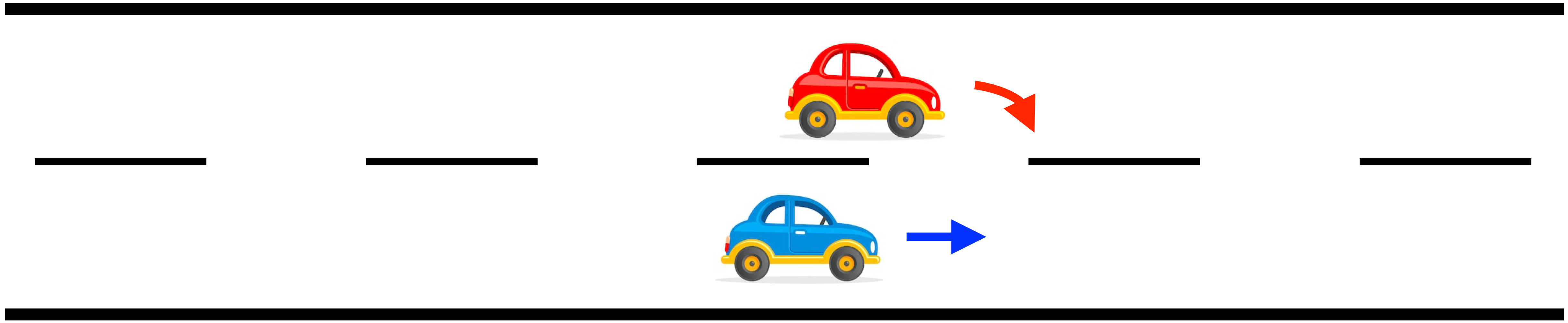
Result of hazard mitigation strategy: refined hazard

$$h_{1.2} \equiv \ell_{red} \wedge r_{blue} \wedge d_0 \wedge R_{red} \wedge v_0$$

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

3

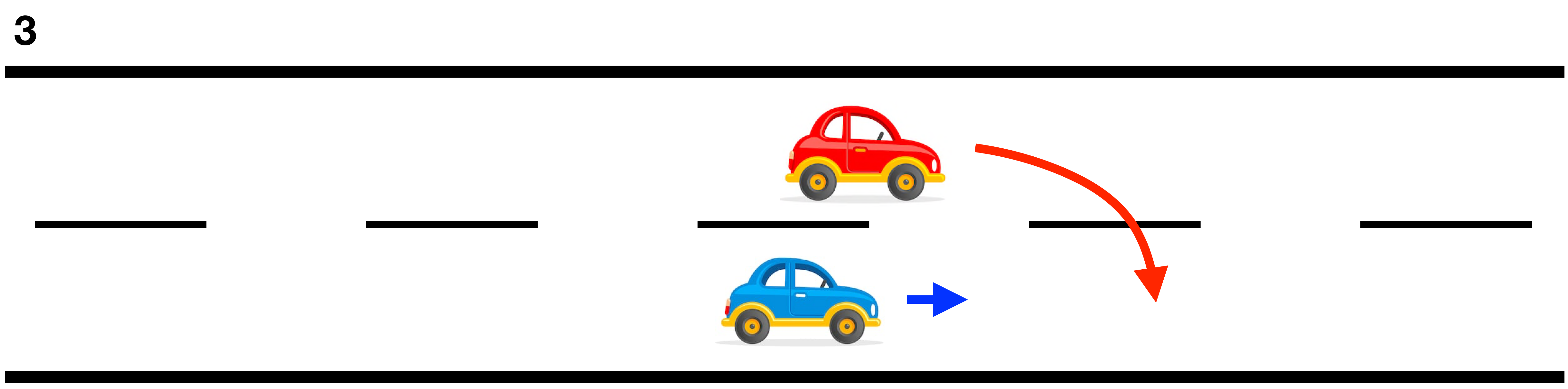


Result of hazard mitigation strategy: refined hazard

$$h_{1.3} \equiv \ell_{red} \wedge r_{blue} \wedge d_0 \wedge R_{red} \wedge v_+$$

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_+

$\ell_{blue}, \ell_{red}, r_{blue}, r_{red}, s_{blue}, s_{red}$

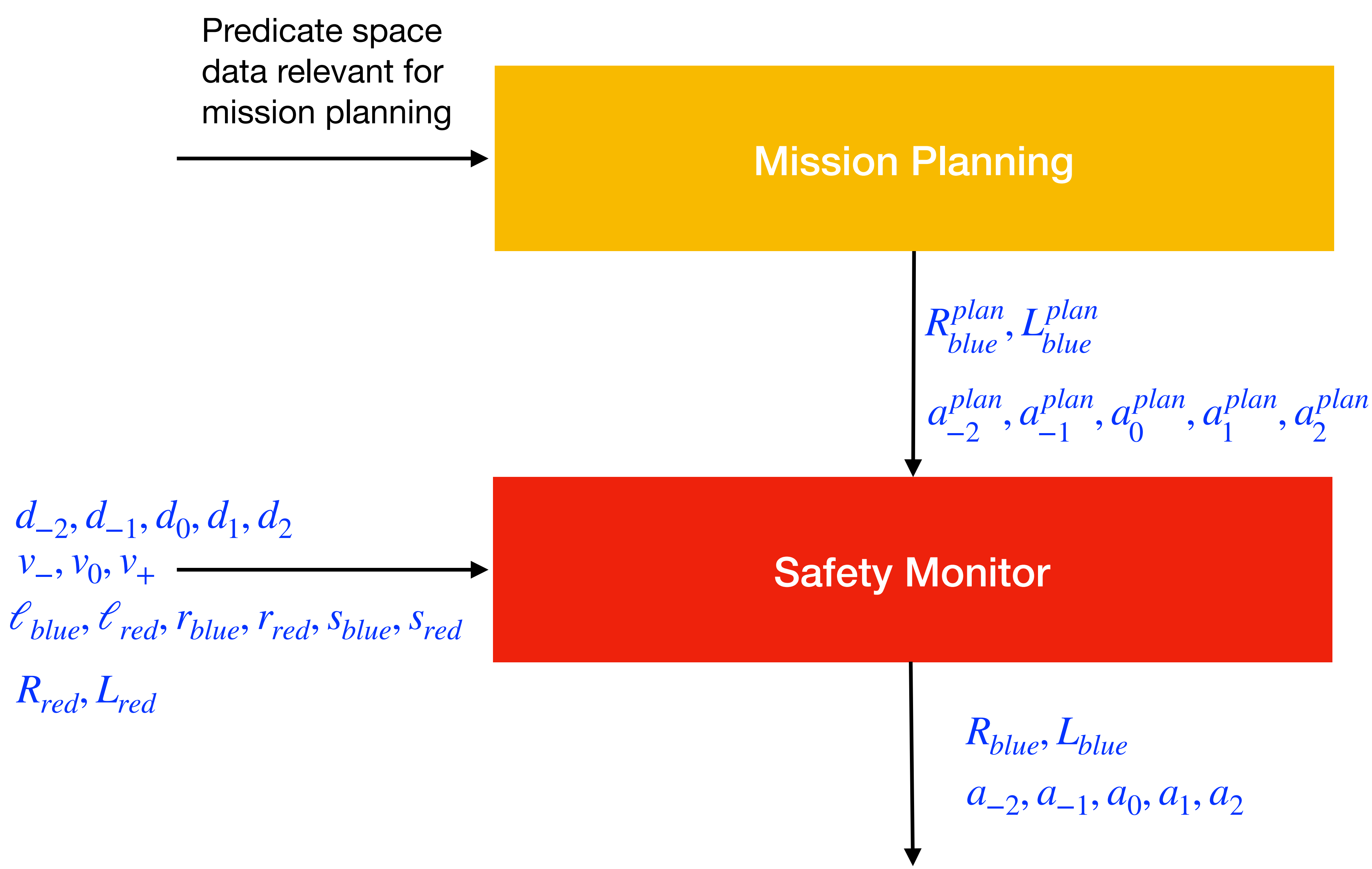
R_{red}, L_{red}

Outputs of safety monitor – from predicate state space

R_{blue}, L_{blue}

$a_{-2}, a_{-1}, a_0, a_1, a_2$

Interplay Between Mission Planning and Safety Monitor



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)
```

```
NEAR(vRel) = l_blue -> Scenario2
            []
            . . .
            []
            r_red -> Scenario3
            []
            . . .
            []
            d_minus2 -> FAR(vRel)
            []
            d_minus1 -> NEAR(vRel)
            []
            d_0 -> CLOSE(vRel)
            []
            d_1 -> SafetyMonitor1
            []
            d_2 -> SafetyMonitor1
            []
            v_minus -> NEAR(-1)
            []
            v_0 -> NEAR(0)
            []
            v_plus -> NEAR(1)
            []
            (vRel >= 0) & L_blue_plan -> L_blue -> NEAR(vRel)
            []
            (vRel < 0) & L_blue_plan -> NEAR(vRel)
            []
            R_blue_plan -> NEAR(vRel)
            []
            a_minus2_plan -> a_minus1 -> NEAR(vRel)
            []
            a_minus1_plan -> a_minus1 -> NEAR(vRel)
            []
            . . .
```

```
CLOSE(vRel) = l_blue -> Scenario2
              []
              . . .
              []
              (vRel == 0) & R_red -> (a_2 -> Scenario3
                                      |~|
                                      a_minus_2 -> Scenario4)
              []
              (vRel == -1) & R_red -> a_2 -> Scenario3
              []
              (vRel == 1) & R_red -> a_minus_2 -> Scenario4
              []
              d_minus2 -> FAR(vRel)
              []
              . . .
              []
              v_minus -> CLOSE(-1)
              []
              v_0 -> CLOSE(0)
              []
              v_plus -> CLOSE(1)
              []
              L_blue_plan -> CLOSE(vRel)
              []
              R_blue_plan -> CLOSE(vRel)
              []
              a_minus2_plan -> a_minus1 -> CLOSE(vRel)
              []
              a_minus1_plan -> a_minus1 -> CLOSE(vRel)
              []
              a_0_plan -> a_0 -> CLOSE(vRel)
              []
              a_1_plan -> a_1 -> CLOSE(vRel)
              []
              a_2_plan -> a_1 -> 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

Discussion of the Per-Scenario MBT-Approach

Benefits

- Per-scenario approach simplifies hazard analysis, because the focus is on a restricted scenario instead of a very complex complete system model capturing all relevant traffic states and evolutions

- The well-established complete MBT approach can be applied to testing the safety monitor, just as for “conventional”, non-autonomous systems

Remaining Risks

- The situation analysis might not identify the correct scenario
- This might lead to inadequate hazard mitigation actions

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

Jan Peleska:
Translating Testing Theories for Concurrent Systems.
Correct System Design 2015: 133-151.
doi 10.1007/978-3-319-23506-6_10

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

Marcus Gerhold and Marielle Stoelinga.
Model-based testing of probabilistic systems.
Formal Aspects of Computing, 30(1):77–106, 2018

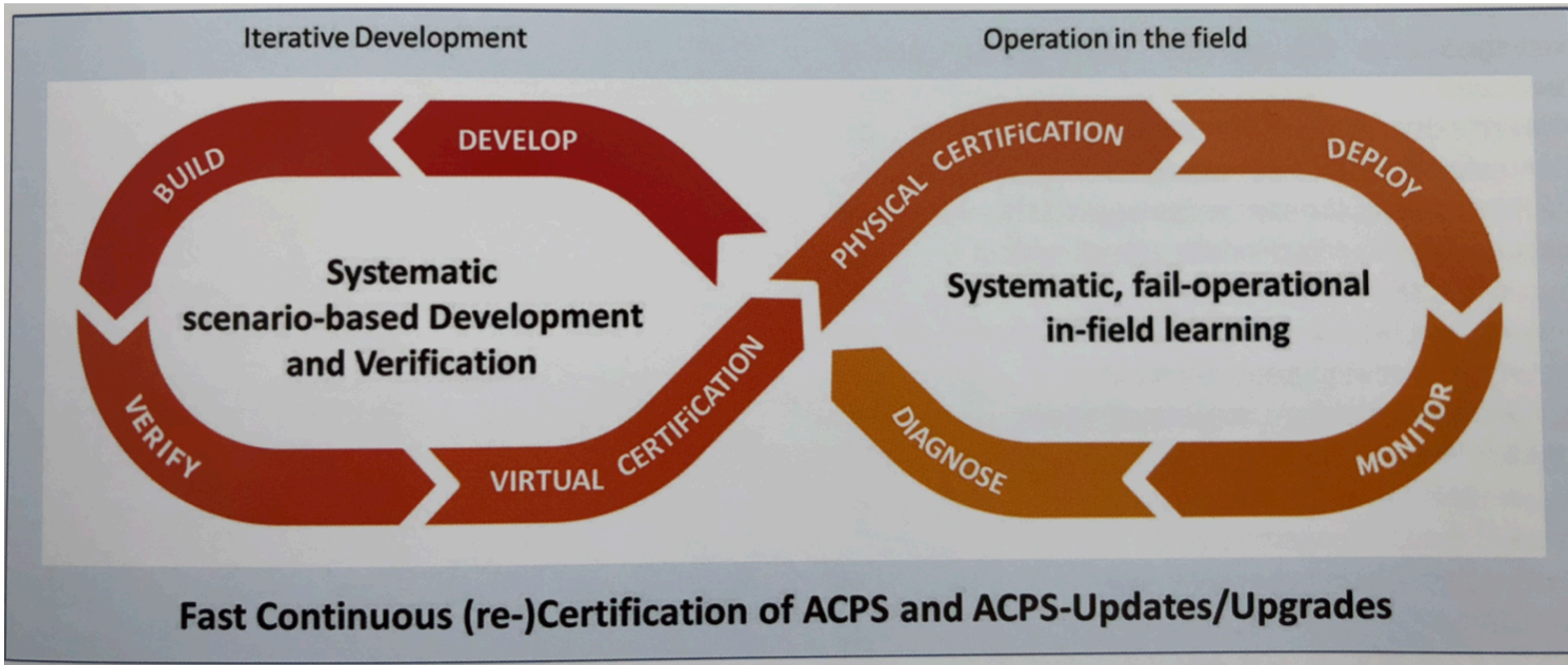
Equivalence Class Testing

- **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

Wen-ling Huang, Jan Peleska:
Complete model-based equivalence class testing for nondeterministic systems.
Formal Asp. Comput. 29(2): 335-364 (2017)

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

- **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

Solved for MBT of safety monitors as described above

This task is easier when focussing on a specific scenario

Allow for testing in simulation environments, performed in the cloud on many CPU cores

Facilitated by predicate abstraction

Facilitated by predicate abstraction

Apply statistic testing

Apply statistic testing

- 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

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!