A Concept of Scenario Space Exploration with Criticality Coverage Guarantees*

Extended Abstract

Hardi Hungar

Institute of Transportation Systems, German Aerospace Center, Lilienthalplatz 7, 38108 Brunswick, Germany
hardi.hungar@dlr.de

Abstract. Assuring the safety of an automated driving system is difficult, because a large, heterogeneous set of traffic situations has to be handled by the system. Systematic testing of the full system at the end of the development seems necessary to be able to reach the required level of assurance. In our approach, the set of potentially relevant, concrete test cases result by parameter instantiation from finitely many more abstract, so called logical scenarios. For nearly all interesting automation systems, even virtual testing via simulation can cover only a tiny fraction of this set of concrete test cases.
Here we present an approach by which a selection of test cases can be shown to be sufficient to assert the system's safety. For that, we make reasonable assumptions about the system's inner workings, and about the way safety of a traffic situation can be captured mathematically. Based on these assumptions a criterion for test coverage is derived. This criterion can be used in a simulation procedure exploring the scenario space as a stop condition. If some additional conditions are met, the criterion is shown to imply sufficient coverage to assert safety of the system under test. We discuss the extent and limitation of the resulting guarantee. We plan to elaborate, implement, and demonstrate this procedure in the context of research projects which develop and apply simulation tools for the verification and validation of automated driving systems.

Keywords: safety testing, automated driving system, simulation, scenario exploration.

* This research was partially funded by the German Federal Ministry for Economic Affairs and Energy, Grant No. 19A18017 B (SET Level 4to5), based on a decision by the Parliament of the Federal Republic of Germany. The responsibility for the content lies with the author.

The final authenticated version appeared in: T. Margaria and B. Steffen (Eds), ISoLA 2020, Springer Nature Switzerland AG, LNCS 12478, pp. 293-306, 2020. The version is available online at https://doi.org/978-3-030-61467-6_19

1 Problem statement

It is widely accepted that it is very difficult to prove that an *automated driving system* (ADS) behaves in a safe manner in its entire *operational design domain* (ODD). Even if the ODD is restricted to a comparatively simple one, e.g. highway traffic, the challenge is substantial.
The goal of a safety argument must be to prove an absence of unacceptable risk, i.e., sufficient safety. Such an argument is based on *evidence*. Currently, it is expected that the bulk of the evidence about the safe behavior will come from various forms of testing. How testing can be judged to be sufficient, and how this can be realized, is the main question of this paper. This is not to say that other forms of evidence like results from FMEA (Failure Mode and Effect Analysis), tool qualification, etc. are not important or indispensable.
It is important, however, that in our opinion, the usual approach to develop safety-critical systems according to the ISO 26262 fails here. This approach relies on a full functional system specification being available in an early design stage phase, a thorough criticality analysis identifying most or all sources risk, and a stepwise development where the output of each step is thoroughly verified. An ADS is too complex to realize such a process, at least at the time being. This puts a large burden on late testing for safety, to make up for the uncertainty about risk inherited from earlier development phases.
Main challenges in late testing are

1. All relevant traffic scenarios from the ODD have to be addressed (completeness of the test specification)
2. Testing must cover the scenario space sufficiently well to identify all critical situations (full criticality identification)
3. The test results must enable to quantify the risk coming from the operation of the ADS (risk estimation)

We are concerned with the second challenge, here: criticality identification.
We assume that the first challenge is met by

1. A specification of the scenario space to be covered in the form of *logical scenarios*
2. A mathematical definition of criticality by suitable *criticality indicators*

We do not address the third challenge, either, but merely, with our solution to challenge two, provide an essential precondition to handle that.
A *logical scenario* is a parametric formalization of a set of traffic evolutions around an ego vehicle. The ego vehicle is not restrained in its behavior. Instantiating the logical scenario with a set of parameter values results in a *concrete scenario*. Semantically, a concrete scenario is a function from ego vehicle behavior to a fully defined traffic evolution. I.e., given an ego behavior, a concrete scenario provides trajectories for all participants.
A *criticality indicator* is a function which assigns, to a given traffic situation, a numerical value which is related to the criticality of that situation, i.e., how close it is to an accident.

Since even simple logical scenarios have a large number of concrete instances, and there will likely be a few hundred logical scenarios, the space to be covered by testing is very large. And it is inconceivable to cover this space with proving ground or field tests. The bulk of testing has to be done by simulation. Ignoring the validity problem of simulation, this offers the possibility to

1. Run a very large number of tests
2. Dynamically and flexibly select test cases to optimize coverage

The test object, called *ego vehicle* (E), is, in simulation, a model of the automated vehicle, including all components of the ADS like sensors, sensor fusion, situation evaluation and decision (Highly automated driving function, HADF), actuation, and dynamics control.
We can assume to have some insight into E, but not a full view of its inner state and function. That means, in at least one main instance of a late testing approach, we face a gray-box situation.

Main Goal The main goal is to define a procedure which explores the space defined by a logical scenario and detects all its instances in which the ego behavior results in situations exceeding a given level of criticality.

2 Conceptual approach

2.1 The ego vehicle

For a completely arbitrary ego, the problem stated in the previous section will be unsolvable. If the ego might change its behavior drastically without apparent reason in the external situation, no testing without deep insight into the inner workings can be complete. We must assume certain well-behavior.
As the ego will result from a thorough development geared towards avoiding bugs, such erratic actions resulting from software glitches can be assumed to be controlled. This results from the fact that a development process following the requirements and recommendations from the ISO 26262 as much as possible will be geared towards minimizing software bugs and component failures. System failures from the remaining errors can thus be assumed to be sufficiently rare. This reasoning does not apply to the overall functionality whose adequacy cannot be similarly ascertained.
Also, E should be deterministic, in principle predictable. Its reaction to a certain situation should always be the same, provided its mission is the same and the history leads to a similar assessment of the situation. The situation includes the road layout, furniture, infrastructure (e.g. traffic lights) state, traffic participants and their actions, weather, communication with the environment or other vehicle (V2X/X2V), and state information about the ego vehicle itself. State information might be available to the HADF (velocity etc.) or not (sensor malfunction). The mission of an ADS is given by route settings and other options like driving style etc. Situation assessment might depend on some limited history, to e.g. account for traffic participants which have temporarily left

the field of view of the sensors. In the definition of test cases (scenarios), this must be taken care of by including a settling phase at the beginning.
A hybrid automaton should be able to model the behavior of the ego. Such automata can capture a control following the common "sense – plan –act" paradigm. Depending on the degree of precision of the simulation, the modeling of the driving physics will be more or less complex, but never leaving the capabilities of such automata. The automaton itself is *not* assumed to be known. In the light of the first assumption, the discrete conditions will depend in a not-too-fragile way on the external state, mission and other parameters, and scenario events (e.g. component failures).
Assumptions on the ego vehicle We summarize our assumptions on the ego vehicle as follows

1. The ego vehicle E is (sufficiently) free from arbitrary, erratic behavior.
2. The behavior of E is deterministic, depending only on the external situation
3. Test scenarios are constructed so that pre-history has no impact on the behavior of E in the test-relevant scenario part
4. The behavior of E can be modeled faithfully by a hybrid automaton

2.2 Criticality indicators

The situations arising in the operation of the ego shall be evaluated for their criticality. There is no precise, universally applicable definition of criticality. Roughly, an accident has maximal criticality (not taking the severity of its consequences into account), a situation where all participants are well separated (keep their safety distance etc.) is uncritical.

Typical mathematical formalizations of the concept of criticality are function like *time to collision* (TTC) or *time headway* (THw). Such functions capture particular aspects of potential criticality. TTC applies to a situation where two trajectories would cross if both traffic participants keep their current speed and direction. E.g., if car B has velocity v_B , and follows car A (velocity v_A), in the same lane with $v_B > v_A$, and the respective positions are p_A and p_B , then

$$TTC(B,A) = (p_A - p_B) / (v_B - v_A)$$

Time headway applies to the same situation, but does not require $v_B > v_A$. It gives the time gap between A and B.

$$THw(B,A) = (p_A - p_B) / v_B$$

Both functions indicate criticality by a low numerical value – zero means accident (the definition of the gap as the position difference is only adequate for a very abstract modelling of the physical situation). In their raw form they are not calibrated. Both go towards infinity for more and more uncritical situations.
To be able to detect critical situations reliably, we require scaled and calibrated indicator functions. What we would like to have are (piecewise) continuously differentiable functions from the set of traffic situations to the interval [0,1] calibrated as follows:

- $0 < s < h < 1$ with

- 0 absence of criticality
- *s*: significant criticality
- *h*: high criticality
- 1: maximal criticality (accident)

Significant and high criticality should refer to standard detection goals: A *high* criticality shall be detected by the exploration. Any value below *significant* criticality is an indication that a situation is in the standard range and does not warrant further examination. These values enter the completeness criterion and guide the detection activities. There are some approaches to objectify the calibration criticality indicators, in particular in the Swedish Conflict Technique (Várhelyi & Laureshyn, 2018). A future elaboration of the presented exploration procedure will take these into consideration.

Criticality Indicator A *criticality indicator* (CI) is a piecewise continuously differentiable function from all situations in a given scenario space to [0,1], with threshold values $s < h \in (0,1)$.

Given that the scenario space is bounded, a CI is always Lipschitz continuous.

TTC and THw can be combined into such a CI, though this is not trivial. Doing that involves taking the reciprocal values, smoothing, scaling, and a softmax of both input values.

2.3 Scenario space

Scenario according to (Ulbrich, Menzel, Reschka, Schuldt, & Maurer, 2015) is a very general notion. It may best be defined as a description of evolutions of road traffic. One may imagine a scenario as a movie storyboard. There are a number of snapshots which capture important intermediate steps, and evolutions – continuous in nature – between those snapshots. In physical reality, the traffic participants in a scenario follow continuous trajectories over time. A snapshot may be taken at any time, resulting in one particular *situation*. A simulation will compute a discretization of the physical reality, producing a finite number of situations. Besides continuous evolutions, there are discrete events which may have an impact on the participants' behavior.

The degree of precision of a scenario may vary considerably, from very abstract, so-called functional scenarios to fully defined concrete instances, with intermediate "logical" scenarios defining sets of concrete scenarios.

Only part of this broad domain is relevant to our method. We consider:

- Results of simulation runs. In those, all traffic participants including the ego vehicle have produced fully defined shown some behavior. These are *fully-defined concrete scenarios* (FCS).
- Definitions of a simulation run. These are, semantically, functions from ego behavior to FCSs. We call them *open concrete scenarios* (OCS).
- Definitions of *sets* of simulation runs, where the behavior of the non-ego participants is parameterized. These are *logical scenarios* (LS). Given a vector of concrete parameters, a logical scenario yields an OCS. It defines a *scenario space*. In this paper, we consider only continuous parameters like thresholds in triggers and scaling factors for movements.

Formats All these variants of scenarios can be expressed in practice by a combination of the formats OpenDRIVE (ASAM e. V., 2020) and OpenSCENARIO (ASAM e. V., 2020), resp., extensions of them. This has been done in the PEGASUS project PEGASUS (PEGASUS Project), and is continued in SET Level 4to5 and other projects.

Maneuvers OCS and LS make use of pre-defined maneuvers ("change lane", "turn right", "follow lane", "accelerate", "keep distance" etc.) with triggers and adequate detailed parametrization to build up situations for the ego where its fitness is to be tested. A more detailed description of scenarios and their definition is given in (Hungar, 2018).

Test scenarios The scenario space to be explored in testing is given by a number of logical scenario. In the test setting we consider here, the ego vehicle shall be subjected in a fully determined way to all possible evolutions which might come to pass in its ODD. The behavior of all non-ego traffic participants is completely specified in the maneuvers and triggers of the scenario definition. Of course, these triggers and maneuvers may include references to the ego behavior. E.g., some maneuver starts when the distance to the ego has reached a certain threshold, or, a vehicle adjusts its velocity dynamically to arrive at a crossing at the same time as the ego. That is, the scenario definition exerts full control over all aspects except the ego behavior. Other testing approaches, not considered here, might prefer a more indirect specification of the scenarios, for instance by using complex driver models to control other vehicles. Traffic evolutions would likely emerge from the interplay of the actors. We consider such loosening of control not to be too useful for safety testing.

Scenario semantics We assume a fully defined semantics for these three types of scenarios. For FCS, these are essentially timed trajectories within the context of the street scenery for all traffic participants. An OCS denotes a function from ego behavior to a full behavior of all participants. And an LS has further arguments for its set of parameters.

The semantical domains for the three types of scenarios may be presented as follows, leaving out details and assuming, wlog, a certain level of abstraction.

Semantical domains Let TP be a finite set of traffic participants, E $\notin$TP the ego vehicle, and let the following domains be given: TIME $= [0, max] \subset \mathcal{R}$ for time, POS for vehicle positions (including orientation), DYN for vehicle dynamics (e.g. velocity), STATE for control states of elements of TP and E, and PAR for logical scenario parameter values. Then

$$[[\text{FCS}]] \subset \text{TP} \cup \text{E} \to \text{TIME} \to (\text{POS} \times \text{DYN} \times \text{STATE})$$

$$[[\text{OCS}]] \subset (\text{E} \to \text{TIME} \to (\text{POS} \times \text{DYN} \times \text{STATE})) \to \text{TP} \to \text{TIME} \to (\text{POS} \times \text{DYN} \times \text{STATE})$$

$$[[\text{LS}]] \subset \text{PAR} \to (\text{E} \to \text{TIME} \to (\text{POS} \times \text{DYN} \times \text{STATE})) \to \text{TP} \to \text{TIME} \to (\text{POS} \times \text{DYN} \times \text{STATE})$$

Explanations on the semantical domains With the usual isometries, these domains can be presented differently (shuffling of arguments, currying and de-currying).

Similarly, the semantics of an OCS or LS can be viewed as a parallel composition between the ego and the TP parts. A particular important view is

$$\text{ocs}\,(e) \in \text{TP} \rightarrow \text{TIME} \rightarrow (\text{POS} \times \text{DYN} \times \text{STATE}), \qquad (0)$$

for osc $\in$ OCS and a particular ego behavior e $\in$ TIME $\rightarrow$ (POS $\times$ DYN $\times$ STATE). Or, retaining the behavior of E, we might write (abusing notation)

$$\text{ocs}\,(e) \in \text{TP} \cup \text{E} \rightarrow \text{TIME} \rightarrow (\text{POS} \times \text{DYN} \times \text{STATE}),$$

for osc $\in$ OCS and a particular ego behavior e $\in$ TIME $\rightarrow$ (POS $\times$ DYN $\times$ STATE).

Depending on the level of abstraction of the modeling of TP and E, the domain POS may have two dimensions (position on a plane), three (adding orientation), or more (e.g., yaw and roll angles). DYN may include velocities, acceleration and other parameters. These are the most common attributes for car modeling, and they are observable from the outside. We do not include placeholders beyond POS and DYN for more attributes which may be used in more sophisticated simulations.

Semantical Properties (OCS, TP) As with E, we assume that each TP in an OCS can be described by a hybrid automaton. Different from E, we have full access to the internal discrete state of each TP, as this is fixed by the respective state of execution of the scenario. We also know the conditions which govern the discrete state changes of the TP during the execution of a scenario. These are assumed to be boolean combinations of formula literals. A literal is a comparison (<, >, …) between terms denoting continuous numerical functions. The function from TIME to POS is continuous for each TP.

Ego and TP Behavior In the view (0) of the OCS semantics above, the full ego behavior e $\in$TIME $\rightarrow$ (POS $\times$ DYN $\times$ STATE) enters the semantical function as an argument. In practical testing, this behavior is of course produced by the system under test. If we test by simulation, the ego behavior is generated by an executable model of the real system.

Given a logical scenario ls, concrete parameters par $\in$ PAR, and t $\in$ TIME, this model provides the behavior of E in the continuation of the current state reached at time t, namely ls(par)(e)([0,t]). And similarly, the scenario description defines how the TP behave from time t onwards, by interpreting the constructs of scenario language.

(Timed) Trajectory The set of positions assigned to a TP or E in the semantics of an FCS form a *trajectory*. If we include only the planar coordinates, this is a two-dimensional curve. Adding the time coordinate yields a *timed trajectory*.

A criticality indicator at a point in time depends only on the visible behavior. We call it the domain of *(timed) situations*.

(Timed) Situation If we take a snapshot of the evolution of an FCS, and ignore the internal states, we get a *situation*, an element of

$$(\text{TP} \cup \text{E}) \times \text{POS} \times \text{DYN}\ , \text{ or, equivalently}$$

$$(\text{TP} \times \text{POS} \times \text{DYN}) \times (\text{E} \times \text{POS} \times \text{DYN})$$

A *timed situation* adds the current time to the situation.

A situation includes all what is externally observables of the TP and E. Criticality indicators are defined on the domain of situations. Given an FCS, a CI can be viewed as a function from TIME to [0,1].

The completeness criterion will also take the internal states of the TP into account. The reason is their importance for answering scenario coverage. If, during the exploration of an LS, a situation arises which is very close to one already evaluated, it need not be evaluated for its CI value. But to decide whether already all continuations of the new situation have been investigated, one needs to know whether also the scenario is in the same execution state, i.e., whether the discrete state history is the same.

(Timed) Constellation A constellation extends a situation by the sequence of internal states of the TP up the reaching the situation. Constellations are elements of

$$(\mathrm{TP} \times \mathrm{POS} \times \mathrm{DYN} \times \mathrm{STATE}^{*}) \times (\mathrm{E} \times \mathrm{POS} \times \mathrm{DYN})$$

A *timed constellation* additionally includes the current time.

The STATE components in (timed) constellations are discrete. We partition the constellations into *clusters*, those subsets sharing a common STATE history. Within a cluster, the semantical function from parameters of an LS to the resulting (timed or not) trajectories would be continuous, if the STATE component of E was known. This results from the role of parameters in the hybrid-automata view of the TPs and the respective assumption on E. Also, a CI would be a continuous function of parameters and time. Since the internal state of E is not assumed to be observable, we must subdivide clusters into *subclusters* to arrive at continuous relations from parameters to observables.

(Timed) Cluster Given $\mathrm{ls} \in \mathrm{LS}$ and the behavior e of the ego vehicle E, the set of *(timed) clusters* of ls(e) is constructed by (a) partitioning the set of *(timed) constellations* into the sets of (timed) constellations sharing the same STATE^{*} values, and (b) projecting the classes to their (timed) situation components (i.e., dropping the STATE^{*} component).

(Timed) Subcluster Each (timed) cluster can be partitioned into a finite set of *(timed) subclusters* s.t. each (timed) subcluster scs is the image of a connected subset PT of $\mathrm{PAR} \times \mathrm{TIME}$ under the semantic function ls(e), and ls(e) is continuously differentiable on PT. *Closed (timed) subclusters* result from topologically closing the (timed) subclusters. clsr(scs) denotes the closure of a subcluster scs.

Properties of clusters and subclusters The existence of the partitioning into (timed) subclusters follows from the finiteness of the discrete control states of the ego vehicle: In a region where the hybrid automata producing the behavior of TP and E stay in the same discrete state, the evolutions are continuously differentiable. The number of subclusters is related to number of discrete states of the ego vehicle. Clusters and subclusters are bounded subsets of a vector space over $\Re$. The closure operation adds the boundary to subclusters.

A situation may belong to several clusters, depending on the history how the situation came to pass.

2.4 Problem statement

We can now restate the exploration problem in a more precise form. Given are

- A logical scenario ls ∈ LS with parameter space PAR
- An ego vehicle in the form of a model or program e
- A criticality indicator c

Then

- H = {v ∈ ls (PAR) (e) (TIME) | c (v) ≥ h } is the set of *highly critical situations for* ls (e).
- S = { v ∈ ls (PAR) (e) (TIME) | c (v) < s } is the set of *insignificantly critical situations for* ls (e).

Goal: The exploration shall compute supersets $H^+ \supseteq H$ and $S^+ \supseteq S$ s.t.

$$H^+ \cap S = \varnothing \text{ and}$$

$$S^+ \cap H = \varnothing$$

Thus, H^+ over-approximates H, but does not include situations of insignificant criticality. Similarly, S^+ over-approximates S, but does not include highly critical situations. H^+ and S^+ might overlap, their intersection is a subset of points which are significantly critical, but not highly critical.

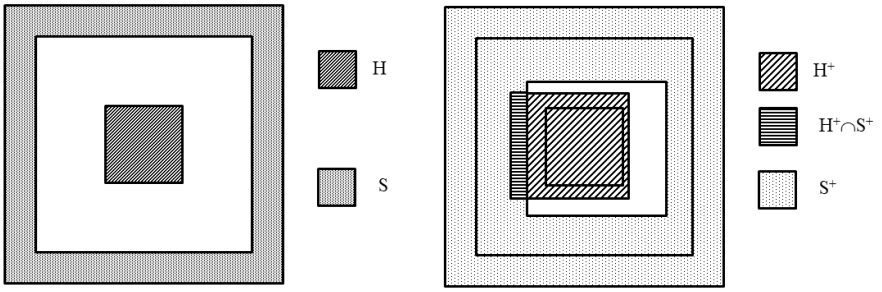

Figure 1: Illustration of the coverage goal

Remark: Obviously, one might remove the intersection $H^+ \cap S^+$ from both sets, once they are computed. For simplicity, we do not include this step in the coverage requirement.

2.5 Simulation

The functionality of a simulation tool is to interpret a fully defined concrete scenario ls (e) (par) over time. The tool computes a discrete approximation of its semantics, which consists of a set of timed trajectories, one for each TP and E. We assume that the simulation operates with a fixed time step size. Since the maximal run time of a scenario

is limited, there are a finite number of situations produced by each run. The situations occurring in such an approximation are witnesses of the semantics of the scenario.

Witness A timed situations w= ls (par) (e) (t) generated by a simulation tool in the simulation of an instance OCS of some logical scenario ls(e), combined with the subcluster scs(w), to which this situation belongs, is called a *witness* of ls(e). If a set of witnesses *W* is given, then

$$W_h = \{w \in W \mid c(w) \geq h\} \quad \text{and} \quad W_s = \{w \in W \mid c(w) < s\}$$

are the subsets of highly critical, resp., insignificantly critical witnesses in W.

The goal of the exploration of the parameter space of a scenario ls(e) is to find a witness for the criticality of every critical point of ls(e), and witnesses for the uncriticality of points which are not near any critical point.

2.6 Coverage Criterion

Let ls, e, and c be given as in the section "Problem Statement" above. The coverage criterion employs the Lipschitz continuity of c on every compact subset of the set of situations. In a given spatial context, let λ be the respective Lipschitz constant.

Lipschitz Neighborhood For a situation v= ls (par) (e) (t), a λ-Lipschitz-neighborhood is a subset of the closure of its subcluster scs(ls (par) (e) (t)) s.t. the criticality indicator is λ-Lipschitz on the neighborhood. A λ-neighborhood is denoted by λ-N(v). A λ-neighborhood containing only points of distance less than d is denoted as λ-N_d(v).

(Vicinity) Coverage Criterion A set of witnesses W *covers* the scenario ls(e) if

1. For each highly critical point (ls (par) (e) (t)) $\in$ H there is a witness w $\in$ W_h and some λ-N(w) s.t. .

$$(ls\ (par)\ (e)\ (t)) \in \lambda\text{-}N(w) \tag{1a}$$

and

$$c(w) - \lambda\ dist(w\ ,\ (ls\ (par)\ (e)\ (t)) > s \tag{1b}$$

2. For each insignificantly critical point (ls (par) (e) (t)) $\in$ S there is a witness w $\in$ W_s and some λ-N(w)s.t.

$$(ls\ (par)\ (e)\ (t)) \in \lambda\text{-}N(w) \tag{2a}$$

and

$$c(w) + \lambda\ dist(w\ ,\ (ls\ (par)\ (e)\ (t)) < h \tag{2b}$$

The criterion distinguishes highly critical and insignificantly critical points by witnesses in their neighborhood. This motivates the name "Vicinity Criterion".

From the sets W_h and W_s and λ-neighborhoods of their elements, one can get H^+ and S^+:

1. H^+ can be made up from the union of some λ-$N_{(c(w)-s)}$(w) of w $\in$ W_h ,
2. S^+ can be made up from the union of some λ-$N_{(h-c(w))}$(w) of w $\in$ W_s

Other criteria which more directly identify regions of interest may be formulated. For instance, one might be interested in a more direct, geometrical representation of an over-approximation of regions of highly critical points. E.g., one might define a concept of *container*, given by a number of defining points. The content of the container is then the convex hull of the defining points. Then, a solution to an adequately reformulated problem would consist in a number of containers whose union contains all highly critical points (and no insignificantly critical ones).

2.7 Exploration Procedure

To make use of the coverage criterion we have to define an exploration procedure which generates witness sets W_h and W_s satisfying the two respective conditions. We sketch a procedure which achieves this goal under the following assumption.

Assumption on simulation adequacy The criticality of the discrete-time FCS computed by the simulation is the same as continuous-time FCS of the semantics.

In practice, one may relax this assumption to permit (more realistically) a small deviation which can be accounted for by adjusting the h and s values.

Exploration concept The witnesses produced by the exploration procedure do not only cover the situation space as required by the criterion. They also cover PAR × TIME densely (in relevant areas). I.e., for some ls (par) (e) (t) in H or S, the respective witness will be ls (par') (e) (t') for some (par',t') with a small distance dist((par,t) , (par',t')) . The exploration starts with a rough sampling of PAR, and goes on to cover clusters and identify subclusters. Clusters are defined by the STATE components of the TP. Subclusters are identified by continuity breaches.

Cluster coverage To cover all clusters resembles the problem of finding input values to activate paths in a program. Indeed, the scenario definition can be viewed a as program. It does not contain unbounded loops, also the maximal number of steps is bounded. Thus, one of the problems of test coverage (infinite number of paths) is avoided. On the other hand, the behavior of the ego is an uncontrollable input. But at least, bounds for the evolution of the ego vehicle are known (max acceleration etc.), and it does not change behavior patterns erratically. Thus, we expect that constraint solving and gradient-directed searches will greatly speed up the process of finding parameter values activating the trigger conditions necessary to visit all reachable clusters.

Subcluster identification Within a cluster, the (differentiable) continuous relation between parameters and situations is only disturbed by the influence of discrete state changes of the ego vehicle. These should be identifiable by systematically validating gradients inside the cluster, i.e., by comparing extrapolated changes in situations (using gradients computed on the neighborhood of a point) with observed changes. Drastic continuity breaches will be easy to identify, others will be discovered during the construction of a covering set of witnesses.

Criticality coverage Hand in hand with subcluster identification, computed situations are analyzed for their witness potential. Given some w = ls (par') (e) (t') with c(w) < s, a neighborhood of (par',t') s.t. its image satisfies condition (2a) and (2b), using conservative approximations of Lipschitz conditions for ls(e). Points of high

criticalities are in the complement of the insignificantly critical ones, and witnesses for those are treated likewise. The search procedure stops when the set of witnesses satisfies (1,a&b) and (2,a&).

Discussion

Assuming adequacy of the simulation and a minimal relevant resolution of parameter values, the coverage problem is finite, but prohibitively large. This means, that any exploration must be incomplete, thus risking to overlook important (highly critical) situations. This work delineates an approach how this deficiency might be overcome. On the one hand, the amount of simulations must not be too large. On the other hand, nothing must be overlooked, or at least, the probability of anything being overlooked must be made very small. We think that the combination of Lipschitz-based extrapolations and systematic, semantics guided search for extrapolation boundaries (i.e. subcluster boundaries) has the potential to solve this problem.

A fine-grained resolution will only be necessary for regions of significant and high criticality, in particular at cluster or subcluster boundaries. Neighborhoods of situations with low criticality will tend to be large. Thus, we expect that relatively few witnesses will cover the vast majority of situations.

We consider it a higher challenge to ascertain that all breaches of continuity will be detected. For that, the assumption on the freedom from erratic behavior of E becomes important. If E might behave arbitrarily, each procedure not covering the full cluster will likely be incomplete. The procedure sketched in the paragraph on subcluster identification above is far from being defined precisely.

There are several ways to technically refine this assumption on the ego, which might help in getting to a concrete procedure definition. One might, for instance, require that state changes are triggered by conditions true for large subsets of the parameter space, if they are not criticality relevant. I.e., only in emergency situations, we have drastic variations in the reactions of E: Those can be found by the detailed coverage of the neighborhood of a critical situation, which is necessary anyhow. The robust conditions triggering changes in uncritical situations would be found by a more superficial parameter variation.

Still, a rigidly provable completeness might be difficult to achieve, even if the conditions on E are tightened. From the outside, the problem of reliable criticality coverage might even appear unsolvable, as it subsumes an instance of model checking of hybrid automata. But the setting of the application is rather specific. The automata producing the behavior of the TP are simple, they do not contain loops. And the actions of the TP on the one side and the ego vehicle on the other are only loosely coupled: The visible behavior on which the traffic participants interact has a high inertia. This is indirectly captured in the assumption on simulation adequacy: A discrete simulation with fixed, discrete step size would not be adequate to compute the effects of, say, a coordinated control of a platoon of cars. Thus, while there is no proof of the usefulness of the completeness criterion, there is also no simple argument for the opposite.

3 Conclusion

We have defined a coverage criterion for a scenario space sufficient to guarantee that highly critical situations and insignificantly critical ones can be reliably distinguished. And we have sketched how an exploration procedure may proceed to achieve this coverage, if a number of assumptions are satisfied[1]. A mathematically precise and practically useful definition of these assumptions will have to be based on practical experiments. From these, one would have to derive a characterization of the specific nature of the components which enables to better delineate the scope of the guarantee assertion.

We plan to go this road. We will implement variants of the exploration procedure sketched above, and try it on sample verification tasks of practical significance. Our hope is that a core method will indeed satisfy mathematically rigorous guarantee criteria. And even if this cannot be achieved, we expect to be able to find lines of argumentation for the significance of the coverage achieved to be able to strongly support a safety case by the simulation findings.

References

ASAM e. V. (12. Mar 2020). Von OpenDRIVE 1.6.0: https://www.asam.net/standards/detail/opendrive/ abgerufen

ASAM e. V. (13. Mar 2020). *OpenSCENARIO 1.0.0.* Von https://www.asam.net/standards/detail/openscenario/ abgerufen

Hungar, H. (2018). Scenario-Based Validation of Automated Driving Systems. *Proc. Int. Symp. ISoLA, LNCS 11246*, S. 449-460.

PEGASUS Project. (kein Datum). Abgerufen am 15. Jun 2020 von https://www.pegasusprojekt.de/

SAE International. (2018). Surface Vehicle Recommended Practice: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On_Road Motor Vehicles J3016-Jun2018.

Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., & Maurer, M. (2015). Defining and Substantiating the Terms Scene, Situation and Scenario for Automated Driving. *IEEE International Annual Conference on Intelligent Transportation Systems (ITSC).*

Várhelyi, A., & Laureshyn, A. (Apr 2018). *The SwedishTraffic Conflict Technique v. 1.0.* Lund University, http://www.tft.lth.se/fileadmin/tft/images/Update_2018/Swedish_TCT_Manual.pdf

[1] It may be noted that the validity of the simulation results themselves is outside the scope of the guarantee. A fortiori, the assumption on simulation adequacy is even strengthening the validity requirement.