Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping Study

Xinhai Zhang, Jianbo Tao, Kaige Tan, *Member, IEEE,* Martin Törngren, *Senior Member, IEEE,* José Manuel Gaspar Sánchez, *Member, IEEE,* Muhammad Rusyadi Ramli, *Member, IEEE,* Xin Tao, *Member, IEEE,* Magnus Gyllenhammar, Franz Wotawa, *Member, IEEE,* Naveen Mohan, *Member, IEEE,* Mihai Nica, *Member, IEEE,* and Hermann Felbinger, *Member, IEEE,*

Abstract—Scenario-based approaches have been receiving a huge amount of attention in research and engineering of automated driving systems. Due to the complexity and uncertainty of the driving environment, and the complexity of the driving task itself, the number of possible driving scenarios that an Automated Driving System or Advanced Driving-Assistance System may encounter is virtually infinite. Therefore it is essential to be able to reason about the identification of scenarios and in particular critical ones that may impose unacceptable risk if not considered. Critical scenarios are particularly important to support design, verification and validation efforts, and as a basis for a safety case. In this paper, we present the results of a systematic mapping study in the context of autonomous driving. The main contributions are: (i) introducing a comprehensive taxonomy for critical scenario identification methods; (ii) giving an overview of the state-of-the-art research based on the taxonomy encompassing 86 papers between 2017 and 2020; and (iii) identifying open issues and directions for further research. The provided taxonomy comprises three main perspectives encompassing the problem definition (the why), the solution (the methods to derive scenarios), and the assessment of the established scenarios. In addition, we discuss open research issues considering the perspectives of coverage, practicability, and scenario space explosion.

Index Terms—Critical Scenario, Automated Driving, Systematic Mapping Study.

1 Introduction

The long and winding road towards high levels of automated driving is fascinating and highly representative of an ongoing technological paradigm shift, see e.g., [1] and further references therein. The past years have seen enormous amounts of funding going into a new automated driving ecosystem of startups, new players and the existing automotive industry, [2], attracting in excess of 80 billion US dollars during the period 2014 to 2017 [3], with likely more than 100 billion US dollars invested the past decade, [4]. As a result, impressive advances have been made and technology has matured. However, the resulting complexity still poses formidable socio-technical challenges for introducing automated driving on a larger scale.

One key challenge refers to reasoning about what can happen during automated driving with relevance for safety engineering activities, from design to assurance. The complexity and uncertainty of the driving environment, and the complexity of the driving task itself, imply that the number of possible scenarios that an Automated Driving Systems (ADS) or Advanced Driving-Assistance Systems (ADAS) may encounter is virtually infinite. It is clear that traditional mileage validation is infeasible for the safety validation of ADS and ADAS, and moreover that ”driven miles” and disengagement reports are far from sufficient for reasoning about risk, see e.g., [4], [5]. This has led to a strong interest in virtual evaluation and verification, allowing for ”safe testing” and exploration, and potentially for cost-efficient assurance based on the created evidence. However, this still begs the following questions: Assuming you have a faithful simulation environment (a challenge on its own), what tests are you to select (i.e. which are critical), how do you derive them, and how do you reason about coverage and the completeness of the safety analysis?

A typical approach to limit the potentially infinite scope of what an ADS may encounter, including unknowns and uncertainty, is to limit the Operational Design Domain (ODD), i.e., the operating conditions under which a given ADS or one or more of its features are designed to function, [6]. This is indeed the most common approach taken for current roll-outs of ADS at high levels of automation. A complementary proposed approach is to attempt to define and limit the behaviors to be encountered on the roads, see e.g., [7]. While this may be very promising in the long term, it is hard to accomplish in the short term given mixed-mode traffic (e.g., manually driven cars together with vehicles at level 4 of automated driving), the complexity of automated driving, and the need to agree on what represents a suitable risk level (compromise between performance and safety).

- *X. Zhang is with Sigma Technology Consulting AB and the autonomous group of Scania CV AB in Sweden. He was with the Mechatronics division of KTH for most of the time when the paper was writing. E-mail: xinhai@kth.se*
- *J. Tao, M. Nica and H. Felbinger are with AVL.*
- *K. Tan, M. Törngren, J. Gaspar, M. R. Ramli, M. Gyllenhammar and N. Mohan are with the Mechatronics division at KTH, Stockholm, Sweden*
- *X. Tao is with the Integrated Transport Research Lab (ITRL) of KTH.*
- *M. Gyllenhammar is with Zenseact AB, Gothenburg, Sweden*
- *F. Wotawa is with the CD Laboratory for Quality Assurance Methodology for Autonomous Safety Critical Systems (QAMCAS), Inst. for Software Technology, TU Graz, Graz, Austria*

A third avenue would be to leverage collaborating driving and smart infrastructure, however this requires multi-stakeholder coordination, infrastructure investments and dealing with new security related threats, thus complicating and delaying such an approach, see e.g., [4].

In any case, it becomes essential to be able to reason about the identification of scenarios - as ”the temporal development between several scenes in a sequence of scenes”, [8]. Scenarios could be seen as ”test cases” within an ODD. As such, scenarios are tangible and concrete and support ADS design and evaluation, with the key challenge that the number of possible driving scenarios that an AD or ADAS may encounter is virtually infinite, and correspondingly for covering a given ODD. In particular, for the purpose of supporting safety engineering, it becomes imperative how ”critical” scenarios are identified, motivated and used. With ”critical” scenarios we refer to situations which cause potential risk of harm (safety risks), that need explicit consideration for risk investigation and potential risk reduction measures (note: we summarize the key terms used in this paper in Section 2).

It is thus not surprising that scenario-based approaches are receiving a huge amount of attention in research and engineering regarding the development, verification and validation (V&V) of ADAS and ADS.

The area is very active in terms of research, standardization, and engineering, encompassing, for example, scenario and ODD modeling, exchange formats, methods for scenario generation, and scenario space exploration, scenario catalogs and standards/guidelines, see e.g., [9], [10], [11], [12], [13], [14], [15].

During our research and work, we also discovered that many different approaches are taken for how to reason about scenarios and critical scenarios, ranging from different abstraction levels, assumptions, methods for scenario generation, exploration, etc. Moreover, we were missing a way to classify the different approaches we found, as a way to organize and compare them, and we were not, to the best of our ability, able to find existing comprehensive surveys on this topic.

This led us to formulate a systematic mapping study focused on Critical Scenario Identification (CSI) methods for ADS and ADAS. Our literature review focuses on approaches to identify critical scenarios to support the development of ADS and ADAS. This, for example, means that scenario modeling languages like OpenScenario are out of the scope of the survey, other than when used as part of a CSI method. The detailed scope of the literature review is described in Section 3.1. Other directions of scenario-based methods are briefly introduced and exemplified in Section 8.

The main contributions of the paper are as follows:

- A comprehensive taxonomy of CSI methods for the development of ADS and ADAS based on a systematic mapping study.
- An overview of the reviewed CSI methods based on the taxonomy.
- Identified open issues and directions for further research.

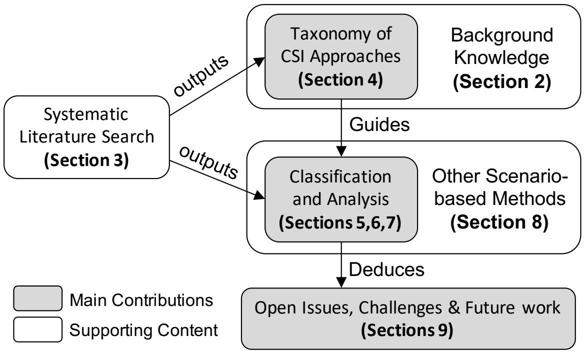
Taxonomy of CSI Approaches (Section 4)
Background Knowledge (Section 2)
outputs
Systematic Literature Search (Section 3)
Guides
outputs
Classification and Analysis (Sections 5,6,7)
Other Scenario-based Methods (Section 8)
Deduces
Main Contributions
Supporting Content
Open Issues, Challenges & Future work (Sections 9)

Fig. 1. Graphical outline of the rest of the paper

The rest of the paper is organized as shown in Fig. 1. Section 2 introduces key concepts, related standards and surveys of relevance for CSI methods for ADS/ADAS systems. Section 3 describes the employed literature review methodology, including the detailed scope, the research questions and the processes to search for relevant primary studies, to analyze the selected studies and to document the findings. The main result of this literature review includes a taxonomy for the studied CSI methods (Section 4) and the corresponding classification of the primary studies (Sections 5, 6 and 7). Other related topics (but outside the scope of this literature review) are identified and briefly introduced in Section 8. Before concluding the paper in Section 10, Section 9 summarizes and discusses all the findings from this systematic literature review, and accordingly suggests future work.

2 Key Terminology and Related Surveys

This section introduces the background knowledge and key concepts/terminology to facilitate the explanation of the taxonomy in Section 4. The definitions of the terms are given in TABLE 1 based on the following standards: SAE J3016 [6], ISO 26262 [16], ISO/PAS 21448 [10], BSI/PAS 1883 [17], UL4600 [11] and ISO 3450 series under development [18], [19], [20]. In TABLE 1, definitions without a citation refer to definitions proposed by the authors of this survey to serve the explanation of the taxonomy. If a definition has multiple citations, it means that the term has been defined in all those citations, and the definition in this paper considers all those definitions.

2.1 ADS and ADAS

We adopt relevant ADAS and ADS terminology from the SAE J3016(TM) ”Standard Road Motor Vehicle Driving Automation System Classification and Definition” [6].

***ADS*:** In J3016, the term ADS (Automated Driving System) is used specifically to describe a highly-automated driving system, which can perform the entire Dynamic Driving Task (DDT) on a sustained basis without human supervision. [6]. Figure 2 shows the main functions of ADS

TABLE 1
Summary of Relevant Definitions

Terminology	Definition
ADAS	The term ADAS (Advanced Driving-Assistance System) is used to describe assistant systems based on Level 3 or active safety systems.
ADS	The term ADS (Automated Driving System) is used specifically to describe a Level 3, 4, or 5 driving automation system.
Active Safety	Vehicle systems that sense and monitor conditions inside and outside the vehicle for the purpose of identifying perceived present and potential dangers to the vehicle. [6]
Safety	The absence of unreasonable risk of harm.
FuSa	Functional Safety: The absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems [16]
SOTIF	Safety of the intended functionality: The absence of unreasonable risk due to hazards caused by the functional limitations of the intended functionality [10]
Harm	Physical injury or damage to the health of persons [16]
Risk of Harm	Combination of the probability of occurrence of harm and the severity of that harm [16]
Hazardous Event	Combination of a hazard and an operational situation [16]
Hazard	Potential source of harm caused by malfunctioning behavior of the implementation of a vehicle-level function [16]
Failure	Termination of an intended behavior of a system due to a fault manifestation [16]
Fault	Abnormal condition that can cause a system to fail [16]
Unintended Behavior	Behavior going beyond the intended behavior of a system due to functional insufficiencies
Functional Insufficiency	Incomplete specification or insufficient implementation of the intended functionality with an unreasonable level of risk [10]
Triggering Condition	Specific conditions of a scenario that serve as an initiator for a subsequent system reaction, possibly leading to a hazardous behavior [10]
Safety-Critical Operational Situation	Traffic conditions (within the ODD), where a hazard is very likely to propagate to a harm [10], [16], [21]
Influential (Scenario) Factors	A parameterized scenario factor (e.g., sun angle or road friction), which may affect the performance of at least one AD function.
Scenario	The temporal development between several scenes in a sequence of scenes. [8]
Scene	A snapshot of the environment including the scenery and movable objects, as well as all actors' and observers' self-representations, and the relationships among those entities. [8]
Critical Scenario	Scenarios that cause potential risks of harm, which need explicit consideration for risk investigation and potential mitigation measures
CSI Method	Methods to find triggering conditions, safety-critical operational situations, or combinations of the two that will lead to harm
ODD	Operational Design Domain: Operating conditions under which a given ADS or AD feature thereof is specifically designed to function [6]. It contains the set of all the influential factors and the possible combinations of these factors.
Functional Scenario	Scenario space representation on a semantic or a high level of abstraction via linguistic notations [22]
Logical Scenario	Scenario space representation on a state-space level with parameter ranges [22]
Concrete Scenario	A concretization of a logical scenario with concrete parameter values [22]

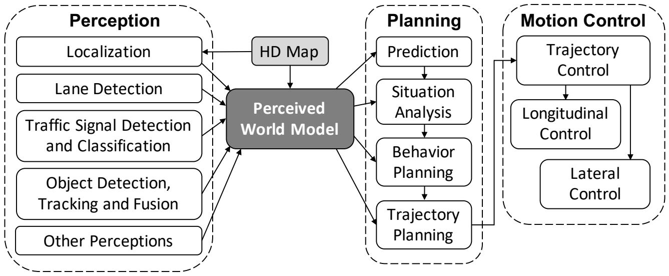

Fig. 2. Nominal AD functions according to Autoware [24], Apollo [25] and Elektrobit open robinos [26]

according Autoware[1] [23], [24], Apollo[2] [25] and Elektrobit open robinos[3] [26].

ADAS**:** The term "Advanced Driver Assistance Systems" (ADAS) is defined in J3016 [6] to describe a broad range of features. ADAS system also includes the *active safety systems*, which are excluded from the scope of this driving automation taxonomy because they do not perform part or all of the DDT on a sustained basis, but rather provide momentary intervention during potentially hazardous situations, such as lane keeping assistance (LKA) systems and automatic emergency braking (AEB) systems.

In our paper, we used the definition of levels of automation from the SAE J3016 [6] to classified the systems in this survey into L3+ (with a human driver), L3- (without a human driver), and active safety system (no continues control of the vehicle). Detailed definitions of these three classes are given in Section 5.1.1.

2.2 Safety and the Sources of Harm

In this paper, safety is considered as a combination of Functional Safety (FuSa) [16] and the Safety Of The Intended Functionality (SOTIF) [10] within a specified operational design domain (ODD) [6]. According to ISO 26262 [16] and ISO/PAS 21448 [10], FuSa and SOTIF have the goal to mitigate residual risk associated with different sources of harm. Extending the discussion in [27], Fig. 3 illustrates the potential sources of harm considered in this paper.

FuSa considers the malfunctioning behavior caused by systematic faults and random hardware faults. It assumes that the design intent of the system is safe. In other words, it assumes that if the vehicle implementation perfectly follows its specification, the vehicle is safe. However, this assumption is not always valid for ADS for the following two reasons:

Reason 1: Due to the openness of the driving environment, the ADS may be exposed to a nearly infinite set of environmental conditions. Therefore the specifications of AD functions may not be sufficient (i.e., not all the conditions are completely considered, e.g., an incomplete traffic sign beside the road). Consequently, some unidentified conditions may lead to safety problems.

Reason 2: Deviation from the intended functionality can also stem from inevitable performance limitations of some

[1] Autoware: https://www.autoware.auto/
[2] Apollo: https://apollo.auto/developer.html
[3] Elektrobit open robinos: https://www.elektrobit.com/products/automated-driving/eb-robinos/

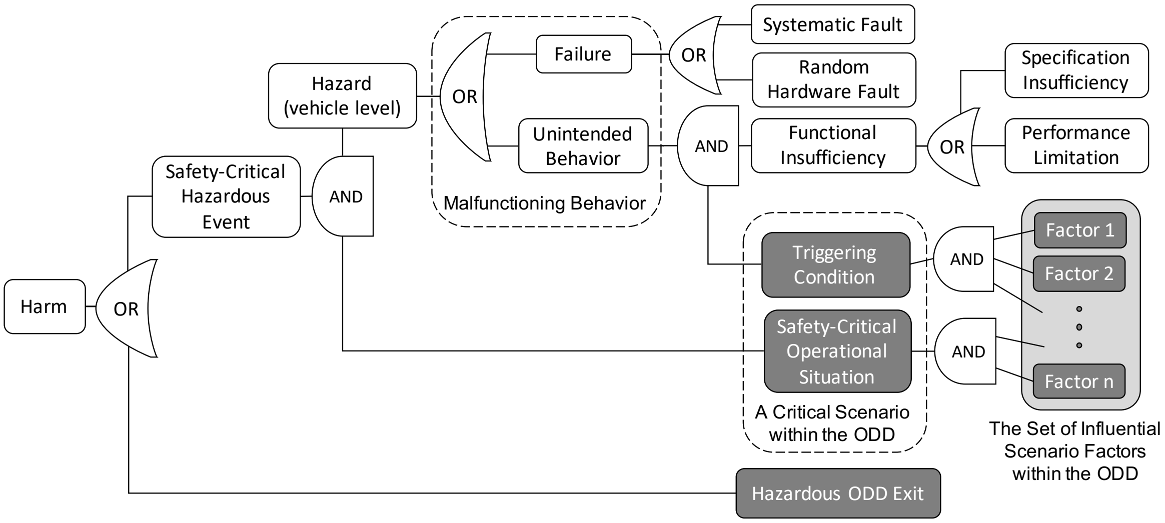

Fig. 3. Sources of harm according to ISO 26262, ISO/PAS 21448 and SAE J3016

advanced functions due to the complexity or the inductive nature (e.g., learning-based methods) of the employed algorithm.

To this end, SOTIF complements FuSa by considering the malfunctioning behaviors stemming from functional inefficiencies, which include the aforementioned insufficient specifications and performance limitations. As shown in Fig. 3, a functional insufficiency will lead to an unintended behavior of that function (e.g., mis-detection of the front vehicle) under a certain triggering condition (e.g., glare). If this unintended behavior is not resolved by the resilience of the downstream functions (e.g., object tracking and sensor fusion) [21], it may propagate to a vehicle level hazard (e.g., a failure to detect a pedestrian implying that braking is not initiated in time).

A hazard may propagate to harm if it occurs in a safety-critical operational situation (e.g., the ego vehicle is running on a highway with a small distance to the front vehicle). For ADS with lower levels of automation, this propagation also depends on the reaction of the involved person (e.g., the driver of an L2 ADS).

Note that the propagation from an unintended behavior to a vehicle-level hazard is not necessarily deterministic owing to the resilience of the system. For example, if the camera-based vehicle detection function fails to detect the front vehicle at one frame because of a sudden glare, this unintended behavior will very likely be fixed by the downstream object tracking function. In addition, hazards stemming from the triggering conditions may not lead to a harm without a safety-critical operational situation.

Another source of harm considered in the study [27] is the misbehavior of other traffic participants. In our framework, an ADS should be able to tolerate a certain level of misbehavior of other traffic participants. The set of all tolerable behaviors should be covered in the (functional) ODD. If an ADS could not survive a tolerable misbehavior of another traffic participant, it will be considered as a functional insufficiency. [4] If the other traffic participants behave beyond the ODD and cause a safety problem, it will be considered as a hazardous ODD exit. However, as far as the authors know, there is no method to completely define an ODD, therefore the boundary between functional insufficiency and ODD exit might be vague.

2.3 Scenarios

The term ”scenario-based testing” was first applied to the development of software systems. [28] A standardized definition of the term scenario in the context of verification and validation of automated vehicles is introduced in ISO/PAS 21448 [10].

2.3.1 Scenario, Scene and ODD

In this paper, the definitions of scenario and scene are directly reused from the study [8] as follows.

Scenario: “A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions & events as well as goals and values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time.” [8] As assumed in [10] and illustrated in Fig. 3, a scenario can be described by a set of influential factors.

Scene: “A scene describes a snapshot of the environment including the scenery and movable objects, as well as all actors’ and observers’ self-representations, and the relationships among those entities. Only a scene representation in a simulated world can be all-encompassing (objective scene, ground truth). In the real world it is incomplete, incorrect, uncertain, and from one or several observers’ points of view (subjective scene).” [8]

All the relevant scenarios that conform to the same scenario description compose a scenario space. An important scenario space defined in [6] is the Operational Design

[4] If this misbehavior is explicitly specified in the logical scenario, it will be considered as a performance limitation, otherwise it will be considered as a specification insufficiency.

Domain (ODD), within which, the ego vehicle is supposed to drive safely. The definition of ODD is as follows:

ODD: "Operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics". [6] ODD essentially defines the operating environment, for which an ADS is designed.

The Pegasus project [29] presented a 6-layer *scenario description model* to categorize the influential factors for scenario and ODD description. The 6 layers are introduced in TABLE 2. To facilitate the discussion in later sections, an additional layer L0 (Internal Ego-vehicle) is added to describe the internal states of the ego-vehicle and the behavior of the driver. Detailed description of these layers can be found in TABLE 2.

2.3.2 Scenario Representation

Menzel et al. [22] classified scenario representations into three levels of abstraction, namely functional scenario, logical scenario and concrete scenario. Functional scenario and logical scenario describe scenario spaces on two different levels of abstraction, while concrete scenario describes a particular scenario.

Functional scenario: A scenario space representation on a semantic level with linguistic scenario notations. [22] "The vocabulary used for the description of functional scenarios is specific for the use case and the domain and can feature different levels of detail." [22]

Logical scenario: A scenario space representation on a state-space level with parameter ranges in the state space. Each parameter correlates to one influential factor. The parameter ranges can optionally be specified with probability distributions. A logical scenario includes a formal notation of the scenario space. [22] Additionally, "the relations of the parameter ranges can optionally be specified with the help of correlations or numeric conditions." [22] In this paper, it is assumed that a logical scenario cannot fully reflect its corresponding functional scenario since relevant parameters cannot be completely listed.

Concrete scenario: A parameterized representation of a particular scenario. Each concrete scenario is an instantiation of a logical scenario, with a concrete value for each parameter. [22]

According to a concrete scenario, an **executable scenario** can be constructed, which can be either a simulation model or a real test. An executable scene refers to an image from a camera or a point cloud from a LiDAR.

In the rest of this paper, functional scenario, logical scenario and concrete scenario are also used to denote a scenario space or a scenario represented with the corresponding levels of abstraction. Fig. 4 depicts the transitions between the three levels of abstraction, which are defined as follows:

Reasoning: This refers to the methods that reason (inductively or deductively) about critical functional scenarios based on knowledge, experience, and information described in the ODD.

Formalization: Using input from the ODD definition, a functional scenario will be formalized and parameterized to a logical scenario with all the parameter definition and their value ranges. In this paper, even though the formalized logical scenario contains more information than its functional scenario, it represents a smaller scenario space since not all the influential factors may have been identified and considered.

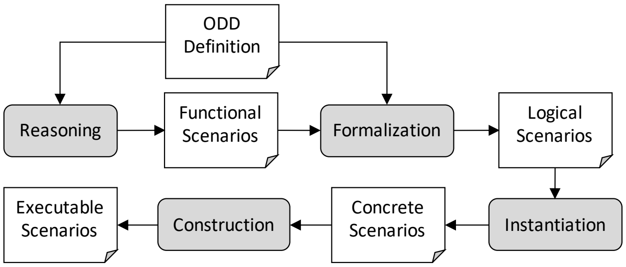

Fig. 4. Relationships between scenario description at different levels of abstraction

Instantiation: In this phase sampling or optimization approaches are used to instantiate the concrete scenario by naive search or guided search methods.

Construction: The concrete scenarios will be converted into executable scenarios with the help of formats like OpenX (OpenDRIVE, OpenSCENARIO) [9], for use in simulators.

Since an ODD is also a scenario space, ODD definitions can also be classified as functional ODD or logical ODD, explained as follows:

- **Functional ODD:** It describes the entire intended ODD on a high level of abstraction. e.g. a particular highway in Stockholm in sunny weather.
- **Logical ODD:** It refers to a parameterized ODD description. It can support the design of the ODD exit detection algorithm. It can also support the formalization of functional scenarios.

Similar as the relation between a functional scenario and its logical scenario, functional ODD represents a bigger scenario space. The misalignment between the functional ODD and the logical ODD constitutes a major source of specification insufficiency. In the rest of this paper, ODD refers to the functional ODD. In Fig. 4, it is the logical ODD that supports the formalization of a functional scenario.

2.4 Critical Scenarios

The concept of a critical scenario is defined in a number of (different) ways in the research literature. Some papers also use related terms (sometimes used as direct synonyms), such as edge case or corner case. This section lists the standard definitions of these synonyms and provides our definition of critical scenario.

Corner case and Edge case: In many cases, corner cases and edge cases are related terms and are often used as synonyms. The probability of occurrence is the most significant difference between them. Corner cases are combinations of normal operational parameters and a rare or unusual condition [10]. Koopman et al. [31] state that not all edge cases are corner cases, and vice versa. Only corner cases with a special combination of conditions that are both uncommon

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

TABLE 2
The 6-layer scenario description model [30]

Layer	Name	Description and Examples
L0	Internal ego-vehicle	Internal states of the other systems of the ego-vehicle other than the SOI + the states or behavior of the driver
L1	Road-level	Road topology, road surface, road elevation (slope, waviness)
L2	Traffic infrastructure	Including traffic rules
L3	Temporary manipulation of L1 and L2	Geometry, topology (overlaid)
L4	Objects	Moving objects (traffic-related objects, e.g. vehicles, pedestrians, movement relative to the vehicle being measured)
L5	Environment	Weather, wind (may affect control), temperature (may affect camera)
L6	Communication	V2X information, digital map

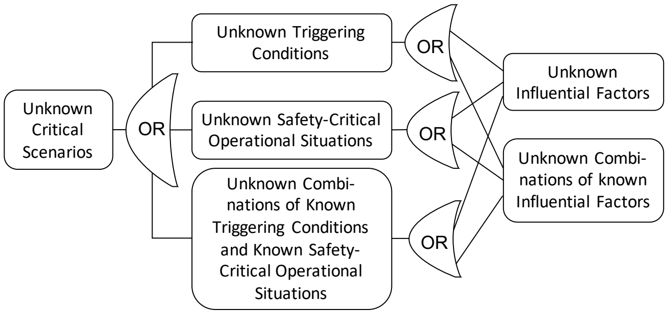
Unknown Critical Scenarios
OR
Unknown Triggering Conditions
Unknown Safety-Critical Operational Situations
Unknown Combinations of Known Triggering Conditions and Known Safety-Critical Operational Situations
OR
OR
OR
Unknown Influential Factors
Unknown Combinations of known Influential Factors

Fig. 5. Sources of unknown critical scenarios

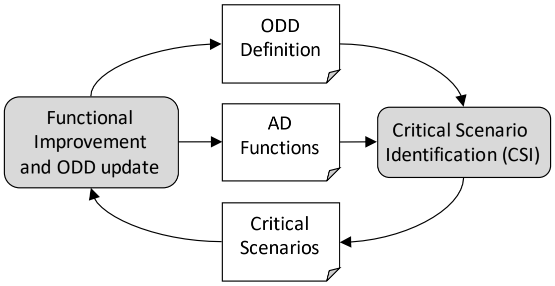
ODD Definition
Functional Improvement and ODD update
AD Functions
Critical Scenario Identification (CSI)
Critical Scenarios

Fig. 6. An iterative process to improve the safety of the intended functionality

and novel are considered edge cases. According to [31], the definition of rare is relative, and it generally refers to situations or conditions that occur frequently enough in a fully deployed fleet to be a problem, but are not documented during the design or requirements process.

Critical scenario: In this survey, critical scenarios are defined as relevant scenarios for system design, safety analysis, verification or validation, with a potential risk of harm. In ISO 26262 [16], risk of harm is defined based on the likelihood of the scenario and the severity of the consequential harm. In this survey, we consider critical scenarios as all the scenarios that may lead to harm. As shown in Fig. 3, a critical scenario may contain a triggering condition and a safety-critical operational situation.

2.5 Critical Scenario Identification Methods

One major goal of ISO/PAS 21448 [10] is to identify unknown critical scenarios, and thereafter make them safe. Triggering conditions and safety-critical operational situations are two major components of a critical scenario within an ODD. Therefore, an unknown critical scenario can stem from either an unknown triggering condition or an unknown safety-critical operational situation. In addition, Annex B2 of ISO/PAS 21448 [10] further assumes that a scenario condition/situation can be modeled as a combination of several influential scenario factors (e.g., heavy rain, glare, slippery road surface, a sudden cut-in of another vehicle, etc.). Under this assumption, an unknown critical scenario can be attributed to either an unknown scenario factor or an unknown combination of known scenario factors.

To this end, Critical Scenario Identification (CSI) Methods are defined as the methods to find triggering conditions, safety-critical operational situations, or combinations of the two that will lead to harm. An ODD definition is considered as the input of a CSI method to delimitate the scenario space.

As shown in Fig. 6, the identified critical scenarios will support the refinement of the automated driving functions to make the ADS safer. They may also help to complete the ODD definition, especially when an identified critical scenario points to an unconsidered aspect of the ODD definition. Meanwhile, functional refinement may also lead to an ODD change, which will initiate a new CSI process in the next iteration.

2.6 Related Survey Papers

While there is a vast amount of literature on various aspects of CSI methods, there are much fewer related survey papers. Other related topics, although outside the scope of this review, are briefly introduced in Section 8.

According to our literature search, to the best of our knowledge, related relevant survey papers can be found in [32] and [33]. Neurohr et al. [32] reviewed and analyzed the literature about the scenario-based testing method for automated vehicles. The authors presented fundamental arguments, principles and assumptions of the scenario-based approach. They also proposed a generic framework (from scenario elicitation to test evaluation) for scenario-based testing and analyzed in detail the individual steps based on the reviewed articles. As a result, they presented various considerations for using and implementing scenario-based testing to support automated vehicle homologation.

Riedmaier et al. [33] performed a survey of scenario-based approaches for safety assessment of automated vehicles. The authors provided an overview of various approaches. They also developed a taxonomy for the scenario-based approach and compared the summarized methods with each other. In the end, this paper integrated the formal

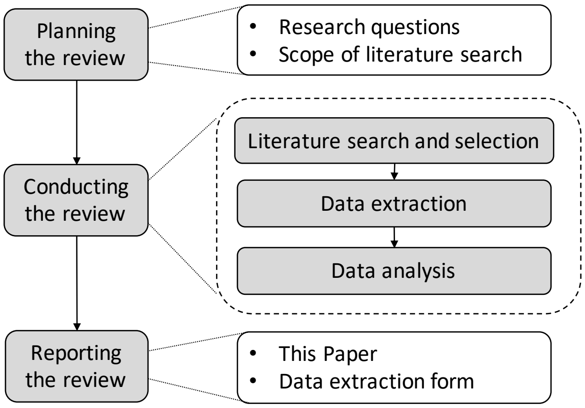

Fig. 7. Overview of the literature review process

verification with the scenario-based approach as an alternative concept.

The differences between this paper and those two survey papers are:

- This paper is dedicated to CSI methods, which is a subset of the focus of the other papers' scenario-based methods.
- This paper provides a systematic mapping study.
- This paper provides a taxonomy for critical scenarios identification approaches.

3 Literature Review Methodology

This literature review follows the guidelines proposed by Keele [34], which divides the whole literature review process into three stages: planning, conducting and reporting. Based on this guideline, our literature review protocol is illustrated in Fig. 7 and detailed in the rest of this section.

3.1 Planning the Review

This section describes the research questions and the planned scope of this literature review.

As discussed in the previous sections, the goal of this paper is to propose a framework to categorize and analyze the CSI methods for ADS and ADAS. Based on this objective, the following research questions are derived.

RQ1: What would be a taxonomy that allows to systematically categorize and compare state-of-the-art CSI methods for ADS and ADAS?
RQ2: What is the current status of CSI methods research with respect to this taxonomy?
RQ3: What are the remaining problems and challenges for further investigation?

The taxonomy in RQ1 can provide a systematic structure of common characteristics to enable the classification and comparison of different CSI methods. With this taxonomy, further researchers or engineers can easily pinpoint a CSI method on the big picture, and thereby understand its strength and limitations. The answer to RQ2 presents the state of the art of CSI methods. It can also help new researchers or engineers in this field have a quick and comprehensive understanding of these methods. RQ3 tries to identify the further directions of this research topic.

To clarify the scope of this literature review, as suggested in [34], the research questions are broken down into individual facets (Population, Intervention, Comparison, Outcomes and Context - PICOC [35]) as:

- **Population:** Software-intensive ADS or ADAS;
- **Intervention:** Approaches applied during development time to identify Critical scenarios;
- **Comparison:** Not applicable;
- **Outcomes:** Improved safety;
- **Context:** Peer-reviewed publications from academia and industry.

The scope of this literature review can be further narrowed down by clarifying the definitions of scenario and criticality. As discussed in Section 2.3.1, the definition of a scenario should follow the one given in [8], and cover at least one layer of the 6-layer model [30]. The included studies must distinguish critical scenarios from other scenarios. General-purpose scenario modeling methods, such as [36], are excluded if they do not consider the identification or generation of critical scenarios. For a similar reason, we also excluded general-purpose data augmentation approaches for training machine learning models (e.g., [37], [38]).

In addition, since CSI methods play an important role for SOTIF [10], this literature review only considers the studies published after the initiation of the SOTIF standard, ISO/PAS 21448 (i.e. 2017).

To this end, the following inclusion and exclusion criteria are formulated. The included papers should satisfy all the inclusion criteria, and should not be covered by any of the exclusion criteria.

I1: Studies describing approaches to identify critical scenarios for ADS or ADAS during development time;
I2: The scenario considered in the study should contain environmental aspects, i.e. content covered by at least one layer of the 6-layer model;
I3: The identified critical scenarios should serve the development of a software intensive system;
I4: Studies published between January 2017 to the end of 2020;
I5: Peer-reviewed studies written in English, and available in full text;
E1: Papers with main focus on cyber security;
E2: Approaches to identify driver misuse scenarios (i.e., unintended usage of the vehicle by human) [10];
E3: On-line methods, e.g., online risk assessment;
E4: Approaches to identify critical scenarios for a hardware component, e.g., radar or LiDAR;
E5: Papers only talking about scenario modeling or the identification of influential factors;
E6: Papers introducing a framework for scenario-based methodology instead of a particular method to identify critical scenarios;

3.2 Conducting the review

As illustrated in Fig. 7, to answer the research questions, three research tasks were conducted. This section provides the details of these three research tasks.

3.2.1 Literature search and selection

The primary studies for this literature review are collected through an iterative process with automatic search and snowballing as shown in Fig. 8. This section describes the details of how we conducted each stage.

Stage 1: A comprehensive initial search string can reduce the number of iterations to determine the final search string as described in **Stage 3**. To better define the initial search strings, an initial set of relevant studies was gathered from two sources. The first source was publications from recent relevant research projects including AutoDrive[5], Prystine[6], Pegasus[7], Enable-S3[8] and AdaptIVe[9]. The second source are the relevant papers included in the two relevant survey papers [32], [33] introduced in Section 2.6. The initial set contains 151 potentially relevant studies. After being filtered by the inclusion and exclusion criteria, 49 studies remained.

Stages 2 and 6: To filter a given set of potentially relevant studies, we reviewed the title abstract and author keywords of each paper with respect to the inclusion and exclusion criteria defined in Section 3.1. After reviewing, each paper was labelled as either *”included”*, *”excluded”* (together with the violated criterion) or *”unclear”*. Unclear studies were further checked by going through their introduction, conclusion and some other parts. When necessary, a discussion among multiple researchers would be conducted to determine the inclusion of an unclear study.

Stage 3: We performed an automatic literature search with the same search string on different databases.

To cover as many relevant studies as possible, we conducted an automatic search on the five major electronic databases on computer science, electronic engineering and automotive technology, namely: IEEE Xplore Digital Library, ACM Digital Library, Scopus, ScienceDirect and SAE Mobilus.

According to the PICOC analysis presented in Section 3.1, the search string is designed as:

$$\text{Search String} = \$AD\ \textbf{AND}\ \$CS\ \textbf{AND}\ \text{”safe*”}$$

where $\$AD$ denotes the identified synonyms of automated driving, and $\$CS$ represents the synonyms of critical scenario or CSI methods. These synonyms are listed in Table 3. $\$AD$ and ”safe*”are searched within the whole text, and $\$CS$ is searched only in title, abstract and author keywords. After stage 7, new synonyms of $\$AD$ and $\$CS$ could be identified to extend the search string. This process was iterated three times until no more synonyms could be found. Table 3 lists the final set of all the identified synonyms. The search result from each database is illustrated in the second column (Found) of Table 4.

Stage 4: Compared to stages 2 and 6, stage 4 had a much larger input set. To share the workload, and also to guarantee the correctness of the filtering, this stage was conducted by two researchers. Since Scopus has overlaps with all the other databases, we assigned one researcher for Scopus and the other for the rest of the databases. The percentage of conflicts (studies that were included by one researcher but excluded by the other) on the overlap was used as a metric to evaluate the explicitness of the criteria and the quality of the filtering. The result of this evaluation is analyzed in Section 3.4.

TABLE 3
Synonyms in $\$AD$ and $\$CS$

	Synonyms
$\$AD$	((automated **OR** autonomous **OR** intelligent) **AND** (vehicle **OR** driving)) **OR** ”self-driving” **OR** ADAS
$\$CS$	((critical **OR** challenging **OR** risky **OR** “high-risk” **OR** hazardous **OR** complex **OR** ”safety-relevant” **OR** accident) **AND** scenario) **OR** ”corner case” **OR** ”falsification” **OR** ((stress **OR** adversarial) **AND** test*) **OR** ”test* scenario” **OR** ”test case”

TABLE 4
Literature search and selection result before snowballing

Database	Found	Selected	Unclear	Included
Initial set	151	49	0	49
ScienceDirect	29	3	2	4
Scopus	261	61	27	67
IEEE	311	35	10	32
ACM	112	1	6	3
SAE	152	14	19	18
Total	**Found: 929**		**Included: 126**	

Each conflict was resolved by a discussion among the two researchers conducting stage 4 and an additional senior researcher.

The result of this stage is listed in Table 4. Studies selected by this stage might also be excluded in **Stage 7**.

Stage 5: Backwards snowballing was conducted in parallel with deep filtering and data extraction. When reading through each study, relevant references were collected. The relevance of each reference was judged by its title and how it is described in the study under review.

Stage 7: The exclusion criteria **E5** and **E6** are sometimes difficult to evaluate by only reading the title and abstract. This stage examines the papers (selected by the previous stages) with a special focus on **E5** and **E6**.

3.2.2 Data Extraction

This task is to answer RQ1. Relevant information needs to be extracted from the primary studies according to a taxonomy. Meanwhile, the taxonomy needs to be updated during the extraction. To start, an initial taxonomy was proposed based on (1) the relevant industrial standards introduced in Section 2, (2) the concepts identified from the initial set of the primary studies, and (3) the previous project experience of the authors. The structure of the taxonomy was inspired by [39]. The initial taxonomy was, thereafter, iteratively updated when reviewing the primary studies, following the process illustrated in Fig. 9. This section elaborates how each stage in this process was conducted.

Stage 1: All the primary studies were thoroughly read by at least one researcher to extract information according to the taxonomy, and to identify new concepts to update the taxonomy.

[5] AutoDrive: https://autodrive-project.eu/
[6] Prystine: https://prystine.eu/
[7] Pegasus: https://www.pegasusprojekt.de/de/
[8] Enable-S3: https://enable-s3.eu/
[9] AdaptIVe: https://www.adaptive-ip.eu/

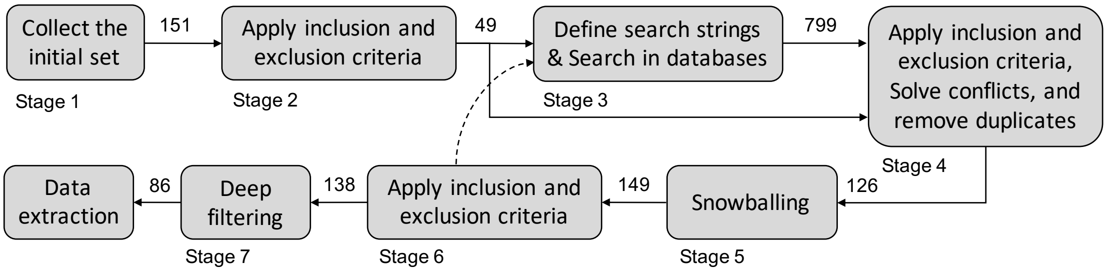
Collect the initial set
Stage 1
151
Apply inclusion and exclusion criteria
Stage 2
49
Define search strings & Search in databases
Stage 3
799
Apply inclusion and exclusion criteria, Solve conflicts, and remove duplicates
Stage 4
126
Snowballing
Stage 5
149
Apply inclusion and exclusion criteria
Stage 6
138
Deep filtering
Stage 7
86
Data extraction

Fig. 8. The stages to collect the primary studies. The numbers on the arrows indicate the numbers of studies given to the next stage.

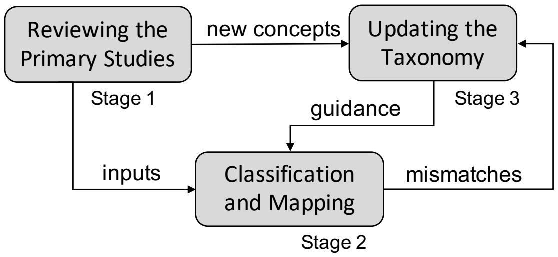

Fig. 9. The stages to extract data from primary studies

Stage 2: As suggested by Keele [34], the extracted information was coded according to the taxonomy and documented in a data extraction form, which can be found in [40]. The result for each study was reviewed by another two researchers to guarantee correctness. For the information that could not be explicitly found in the study, it was marked as "not given" in the data extraction form and if possible, a reasonable assumption was given according to our understanding. Note that the data extraction form [40] includes 91 references in total, where 5 of them are regarded as out of the scope. Thus, primary studies contain 86 papers. A list of acronyms during the systematic mapping study is also included in the data extraction form.

Stage 3: The taxonomy was iteratively updated when (1) a new concept, which had not been included in the taxonomy, was identified from stage 1; or (2) a mismatch (i.e. a study that cannot be reasonably classified by the taxonomy) was detected from stage 2. When the taxonomy was updated, the extracted data would also be updated according to the new taxonomy.

Section 4 presents the final taxonomy, and the data extraction form can be found online as [40]. This task is also the basis of the next task to answer RQ2 and RQ3.

3.2.3 Data Analysis

The goal of this task is to answer RQ2 and RQ3. In the previous task, information was extracted from each primary study according to the taxonomy. This task analyzes the extracted information with systematic and statistical approaches. The analysis results answer RQ2. The findings from the analysis answer RQ3. The analysis results and findings are presented in Sections 5, 6 and 7 and discussed in Section 9.

3.3 Reporting the review

The reporting contains two parts. The first part is this paper, which presents the relevant terminology, the literature review methodology, and the answers to the research questions. The second part is the data extraction form [40], which summarizes each primary study according to the taxonomy.

3.4 Threats to Validity

This section discusses the potential threats to the completeness of the literature search and our mitigation approaches. One of the main threats to the validity of this systematic mapping study is incompleteness. The risk of this threat highly depends on the selected search string, the limitation of the employed search engine, and the quality of the literature filtering (stages 2,4 and 6 in Fig. 8). To reduce the risk of an incomplete search string, the search string was constructed iteratively until no new synonyms of $\$AD$ and $\$CS$ could be added to Table 4. In order to omit the limitations of the employed search engines, multiple search engines were used on different databases with overlaps. To avoid the exclusion of relevant papers during literature filtering, the label *unclear* was introduced for the case where the inclusion of a paper could not be determined with high confidence. This label guaranteed that papers were excluded with high confidence. In addition, the risk of incorrect exclusion could be implicitly evaluated by the number of conflicting papers (i.e. papers that were included by one researcher but excluded by the other) within the overlap between Scopus and other databases. Among the total 77 overlapping papers, there were only 3 conflicts. After further confirmation, only 1 paper (out of 3) was included. In addition, incompleteness is also introduced by the inclusion criteria I4, which excludes papers published before 2017. It is assumed that CSI methods proposed before 2017 are sufficiently studied and improved by later research work. Therefore, missing those early papers will not significantly affect the result of the taxonomy.

Another important threat is the robustness of the taxonomy to describe any CSI method. To guarantee the taxonomy has sufficient concepts to cover all the selected papers, an iterative approach was conducted to update the taxonomy and the extracted data until the taxonomy converged. Furthermore, the capability of the taxonomy to describe a new paper was verified by the 9 papers (after deep filtering) found in the snowballing phase. All these papers were successfully classified by the taxonomy without the need to add any new concepts.

4 Overview of the Taxonomy

Employing the methodology introduced in Section 3.2.2, a hierarchical taxonomy for CSI methods was identified as depicted in Fig. 10.

Inspired by [39], the first level of the taxonomy hierarchy structures the studied CSI approaches with the following three fundamental categories, which reflect the common logic of a complete research.

- ***Problem Definition***: What kind of scenario is being identified? Why are these identified scenarios important? (Section 5)
- ***Solution***: What techniques are applied to identify the critical scenarios? What external information / data is needed? (Section 6)
- ***Evaluation***: How are the validity of the approach and the identified critical scenarios assessed? (Section 7)

Each of these three top-level categories is decomposed into multiple subcategories. Each leaf category in the taxonomy has a number of possible values to categorize CSI methods. The rest of this section introduces all the subcategories.

4.1 Subcategories of *Problem Definition*

The *Problem Definition* category specifies the problem that the approaches aim to solve. It is decomposed into the following subcategories.

Usage of the Scenarios: This category classifies the CSI methods according to how the identified critical scenarios are supposed to be used. Subcategories include 1) the System of Interest (SoI), i.e. the AD system/function whose development is supported by the identified scenarios; and 2) how are the identified critical scenarios used in different development phases. For example, Li et al. [41] proposed a method to find critical scenarios as test cases (*purpose*) for the verification (*phase*) of a whole ADS (*SoI*).

Target ODD: As depicted in Fig. 4, a clear and sufficient ODD definition is necessary for the reasoning of critical functional scenarios and the formalization of a functional scenario to a logical scenario. This category analyzes how ODD is defined and used in each primary study.

Definition of Criticality: The studied CSI methods aim to select or generate critical scenarios, which are distinguished from other scenarios from a specific perspective. Even though the definitions of criticality are not explicitly given in most of the primary studies, this category explicitly classifies the criticality according to the characteristics of the identified scenarios.

Level of Abstraction: This category classifies the studied approaches according to their inputs and outputs in terms of the levels of abstraction of their scenario representation, as described in Section 2.3.2. For example, the approach proposed by Li et al. [41] identifies critical concrete scenarios within a given logical scenario. Therefore this approach is classified as "*logical → concrete*". Similar approaches to identify critical scenarios according to a scenario with a higher level of abstraction are called deductive approaches. In contrast, inductive approaches find critical scenarios based on a set of lower-level scenarios.

4.2 Subcategories of *Solution*

The category *Solution* classifies the primary studies according to how they find critical scenarios within the constructed scenario space. This section explains all its subcategories.

Scenario Space Construction: To systematically identify critical scenarios, scenarios need to be consistently represented to construct a scenario space. This category classifies the CSI methods in terms of 1) ***Content***: what aspects are included in the scenario model regarding the 6-layer model introduced in Table 2; and 2) ***Representation***: how they are represented (e.g., formally, semi-formally or qualitatively).

Scenario Space Exploration: Under this category, the studied CSI approaches are categorized in terms of 1) which algorithm or technique is adopted to explore the scenario space; 2) how the criticality of a scenario is assessed during the exploration; and 3) what mechanisms are used to guarantee or improve the coverage of the exploration. Since the criticality of a scenario can hardly be directly measured, a surrogate measure is commonly used for *criticality assessment*. One common example of a surrogate measure is the time-to-collision (TTC) with the front vehicle in simulation. However, since a surrogate measure cannot completely reflect the criticality of a scenario, as shown in Fig. 11, the set of potential critical scenarios (identified according to a surrogate measure) may slightly mismatch with the set of all the critical scenarios in reality. The *criticality assessment* category collects all the surrogate measures used in the primary studies, and maps them to different criticality definitions.

Required Information: This category summarizes the necessary information each CSI method needs to identify critical scenarios, such as relevant databases, assumed vehicle dynamic/kinematic models, functional models or implementations of the SoI, pre-defined surrogate measures for criticality assessment and environmental information provided by a simulator (e.g., CARLA or PreScan, etc.).

In Section 6, CSI approaches are clustered into several groups according to the similarity between the approaches based on all the subcategories under the *Solution* category.

4.3 Subcategories of *Evaluation*

For the *Evaluation* category of the taxonomy, three subcategories are considered. They are explained as follows:

Approach Verification: This category describes the techniques used to assess the availability and efficiency of the approaches proposed in the primary studies.

Criticality Validation: In contrast to the *Approach Verification*, this category focuses on the evaluation of the identified scenarios. As shown in Fig. 11, identified potential critical scenarios based on surrogate measures may not fully overlap with the set of all the critical scenarios. Reasons for this mismatch include, but are not limited to, the low fidelity of the employed simulator, the assumptions made to simplify the exploration and the limitations of the selected surrogate measure. This category summarizes the approaches to validate the criticality of the identified critical scenarios.

Coverage Assessment: This category enumerates the approaches to assess the coverage of the identified scenarios with respect to the whole scenario space.

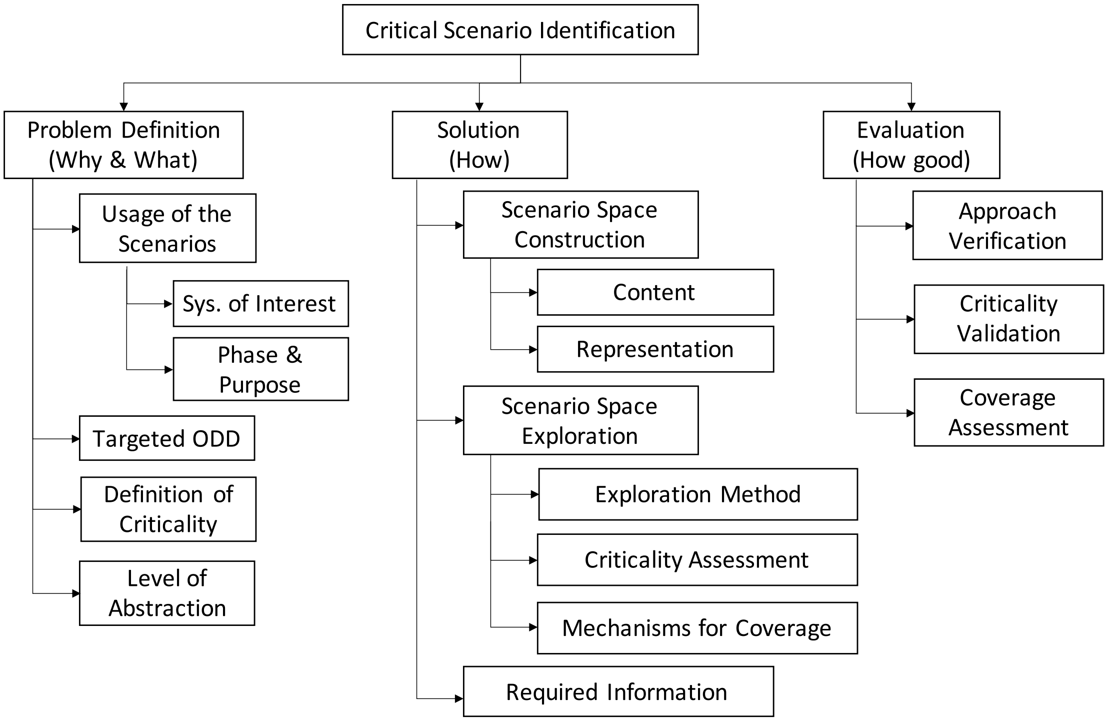

Fig. 10. The taxonomy for critical scenarios identification approaches, achieved from the reviewed literature

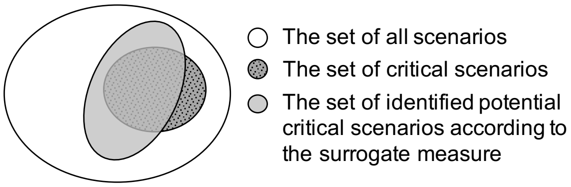
○ The set of all scenarios
● The set of critical scenarios
○ The set of identified potential critical scenarios according to the surrogate measure

Fig. 11. Relationship between critical scenarios and potential critical scenarios identified according to a surrogate measure

5 The Problem Definition Category

This section presents the classification result of the primary studies according to the *problem definition* category of the taxonomy.

5.1 Usage of the Identified Scenarios

In the primary studies, the identified critical scenarios can be used for different Systems of Interest (SoI) under different development phases.

5.1.1 Systems of Interest

Section 2.1 introduces the common functions of ADS or ADAS. As discussed in Section 3.1, the studied CSI approaches are used to support the safety analysis, verification and validation of the whole ADS (or ADAS) or a particular function of the ADS (or ADAS).

Many of the primary studies do not explicitly specify their Systems of Interest (SoIs). In this section, SoIs of the primary studies are classified according to their targeted levels of automation and their functionality.

Section 2.1 introduces how ADS and ADAS can be classified according to the level of automation. Accordingly, the SoIs of the primary studies are classified into the following categories based on the corresponding criteria.

- ***L3+*:** CSI methods will be classified into this category if their SoIs have both longitudinal and lateral control, and they do not require a driver.[10]
- ***L3-*:** CSI methods will be classified into this category if their SoIs have only longitudinal control [42], [43], [44], [45], [46], [47] or a driver's operations [48] are required.
- ***Active Safety*:** CSI methods will be classified into this category if their SoIs cannot provide continuous control of the vehicle [48], [49], [50], [51], [52], [53], [54], [55], [56].

It is assumed that CSI methods for higher levels of automation can also work for lower levels. Therefore, if a CSI method is designed for multiple levels (e.g., the methods to find safety critical operational situations can be used for all the levels), it is classed into the highest level. The statistical result of this classification is shown in Fig. 12 (a).

An SoI can be the entire ADS (or ADAS) or a particular functionality of it. From this perspective, the primary studies are classified into the following categories based on the corresponding criteria derived from Fig. 5 and Fig. 2. The statistical result of this classification is shown in Fig. 12 (b). Detailed classification of all the primary studies can be found in the data extraction form [40].

- ***Perception*:** CSI methods in this category try to find triggering conditions of unintended behaviors (e.g.,

[10] Since most of the primary studies are classified in this category, references to the primary studies are only given for the other two categories. The rest of the primary studies belong to this category.

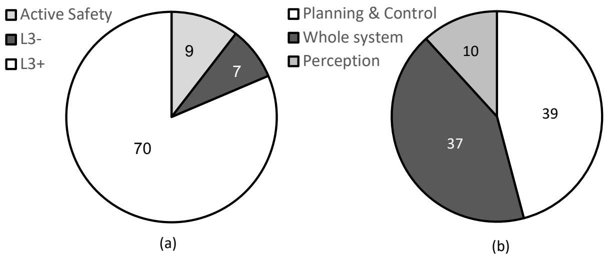

Fig. 12. Number of primary studies in each SoI category

TABLE 5
Relations between SoIs and influential factors on different layers

	L0	L1	L2	L3	L4	L5	L6
Perception	0	0	0	0	0	0	0
Planning and Control	0	5	2	0	37	2	0
The Whole System	3	31	5	0	44	28	0
Total	3	36	7	0	81	30	0

wrong object detection and wrong object classification) of the perception functions. A CSI method will be classified into this category if the inputs of its SoI are the driving environment, and the outputs are the information in the Perceived World Model.

- ***Planning and Motion Control*:** CSI methods in this category try to find triggering conditions of unintended behaviors (e.g., collision or other risky behaviors) of the planning and motion control functions. The input of these functions is the Perceived World Model, and the outputs are the control commands to the actuators (e.g., the steering wheel, the throttle and the brake.) Therefore a CSI method will be classified in this category if it assumes the Perceived World Model is given as ground truth, or ground truth with added noises [57], [58], [59], [60].
- ***The Whole System*:** CSI methods in this category try to find 1) potential safety-critical operational situations; or 2) critical scenarios as combinations of both triggering conditions and safety-critical operational situation. The input of the whole system is the driving environment, and the outputs are the control commands. A CSI method will be classified into this category if it does not assume the Perceived World Model is given and tries to find scenarios that may lead to vehicle level hazards (e.g., unintended brake) or accidents (e.g., crash).

Unintended behaviors of different functionalities may have different triggering conditions attributed to scenario factors on different layers introduced in TABLE 2 [15]. TABLE 5 illustrates the numbers of primary studies that consider influential factors on a particular layer for a particular SoI. Detailed models of these influential factors are analyzed in Section 6.

5.1.2 Phase and Purpose

As illustrated in Fig. 13, in the primary studies, the identified critical scenarios are used in every phase of the development V model. The gray boxes in Fig. 13 list what the identified critical scenarios can support in the corresponding development phase.

Requirement Analysis:

This phase analyzes functional requirements or safety requirements at the vehicle level. In ISO 26262, Hazard Analysis and Risk Assessment (HARA) is an essential step to identify all the potential hazardous events. As depicted in Fig. 3, each hazardous event is a combination of a hazard and an operational condition. With the methods proposed in [30], [61], [62], [63], the identified pre-crash functional scenarios can be used as the set of all the operational conditions for HARA as described in [64].

Most of the studied CSI approaches treat the identified critical scenarios as test cases used in the verification and validation phases. The failed test cases or the identified critical scenarios through simulation can also support the identification of specification insufficiency.

Some of the studied CSI approaches can also support the analysis of influential factors by determining the critical regions (i.e. particular values or value ranges of a set of parameters) where critical scenarios are significantly more probable [57], [58], [59], [60], [65], [66]. The identification of unknown influential factors is out of the scope of this survey but is briefly discussed in Section 8.5.

System Design:

This phase decides the system configuration and the decomposition of vehicle level requirements to component level. However, in this survey, no CSI approach was found to support requirement decomposition. System configuration includes the selection of sensors [55] and sensor ranges [67].

Component Design:

In this phase, the SoIs are certain AD functions, whose requirements are decomposed from the vehicle level. Since different AD functions may be sensitive to different environmental factors, influential factors are commonly analyzed on the component level rather than the vehicle level [15]. One type of approach is to evaluate whether the variance of a particular parameter will significantly affect the criticality of the scenario [68]. For example, if changing the color of a vehicle in front will significantly affect the success rate to detect it, vehicle color will be considered as an influential factor. Critical regions can also be determined for a certain AD function such as the perception [57], and the planning and control functions [47], [57], [69], [70], [71]. In addition, for machine learning based algorithms, identified critical scenes can also be added to the training set to support targeted data augmentation [68]. If the critical scenarios are used as test cases, the failed test cases can help with the identification of functional insufficiency, which includes specification insufficiency and performance limitations.

Component and System Verification:

Nearly 55% of the CSI methods proposed in the primary studies focus on the identification of critical concrete scenarios, which are used in the verification phase as test cases. The generated test cases can be used to verify the whole AD system (e.g., [41]) or a particular AD function (e.g., [42]) in either the real world or a simulated world [72]. Detailed analysis of these methods can be found in Sections 6.1 and 6.2.

System Validation:

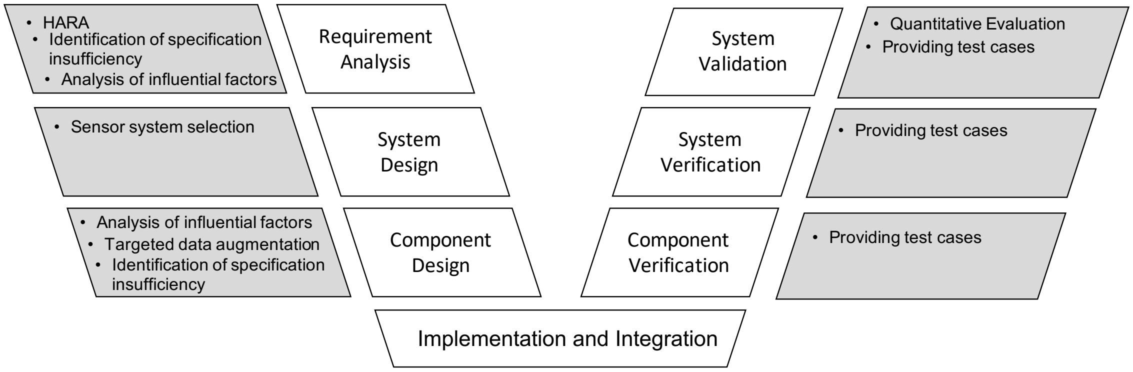

Fig. 13. How CSI methods support different development phases

A common validation method is to estimate the accident rate or failure rate through the Monte-Carlo simulation by randomly sampling the scenario space. Since critical scenarios are relatively rare, Monte-Carlo simulation with a small sample size may lead to a bad coverage of the critical scenarios and hence increase the estimation error. As a variant of Monte-Carlo simulation, importance sampling reduces the estimation error by assigning more samples to the critical but relatively rare scenarios. Therefore importance sampling methods entail the identified critical scenarios or critical regions as input. Li et al. [73] proposed a theoretical foundation of these sample-based validation methods and pinpointed the challenges.

In addition, as discussed in [11], [74], test cases that are potentially critical for most of the ADS implementations can also serve validation purposes, e.g., scenarios with heavy fog or a sudden cut-in in front of the ego vehicle. This type of critical scenario is defined in Section 5.3 as a non-implementation-specific critical scenario. More analysis on how to find these scenarios can be found in Section 6.

5.2 Targeted ODD

As shown in Fig. 4, a clear ODD definition is necessary for the reasoning of critical functional scenarios, and the formulation (together with improvisation) of a functional scenario to a logical scenario. As discussed in Section 3.1, improvisation and formulation are out of the scope of this literature review. Therefore this section only focuses on how ODD supports the reasoning of critical functional scenarios. A clear ODD definition is essential to derive a complete set of critical concrete scenarios. However, none of the primary studies provide an explicit ODD definition. Some of the primary studies explicitly or implicitly provide the scope of the ODD, e.g., driving on a highway [30], [61] and driving on a structured road [62], [63], [75], [76]. Methods in these studies make assumptions about the environment (e.g., the behavior of other vehicles) within the ODD, and propose systematics to reason about critical functional scenarios based on these assumptions.

5.3 Definitions of Criticality

As discussed in Section 2.4, in this paper, critical scenarios are considered to be more important than non-critical scenarios in terms of safety analysis or verification. The definitions of criticality are tightly connected with the usages of the identified critical scenarios. However, most of the primary studies do not provide an explicit definition of criticality. Instead, they explicitly define the employed surrogate measures for criticality. In this survey, these measures of criticality are classified according to two dimensions, as illustrated in Table 6. This classification is also used to specify the definition of criticality in each primary study. The two dimensions are explained in the rest of this section. The surrogate measures are detailed in Section 6. Neurohr et al. [77] provided a more comprehensive discussion on criticality definition and criticality evaluation methods.

The first dimension specifies whether the identified scenarios are implementation-specific. An implementation-specific critical scenario is only determined to be critical for a particular implementation of an AD function or system (set of functions). It may or may not be critical for other implementations of the same function or system. In contrast, non-implementation-specific critical scenarios refer to the ones that are critical for most of the implementations of the same function or system. For example, scenarios with a heavy fog are critical for most of the camera-based object detection/classification functions.

The second dimension classifies criticality according to consequence. Scenarios that are highly likely to cause a hazardous event are defined as safety-critical, while scenarios that may lead to a malfunctioning behavior are classified as function-critical. According to Fig. 3, both safety-critical and function-critical scenarios can support the identification of unknown functional insufficiency and the corresponding triggering conditions. There are also primary studies whose criticality definition includes other perspectives such as comfort and traffic impact. These perspectives are out of the scope of this survey.

Function-critical scenarios can be further classified according to the awareness of the consequential malfunctioning behavior. If the consequential malfunctioning behavior is pre-defined, (e.g., a collision or a misclassification of a certain object), the criticality of the identified scenarios is consequence-aware. Approaches to find consequence-unaware critical scenarios tend to find scenarios where malfunctioning behaviors are tend to be triggered. These scenarios may help to find unknown influential scenario factors [118], [119], [124], [127].

Different primary studies may implicitly adopt different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

TABLE 6
Surrogate measures of each criticality definition. The cluster numbers (C1-C5) are introduced in Section 6

	Implementation-specific	Non-implementation-specific
Safety-Critical	***KPIs in Simulation:*** **C1** [41] [42] [43] [44] [45] [50] [51] [53] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] **C2** [59] [96] **C3** [55] [75] [97] ***Collision:*** **C1** [49] [52] [48] [98] **C2** [46] [58] [60] [99] [100] [101] **C3** [102] [103] [103] [104] [97] **C4** [32] ***Formal Specification:*** **C1** [65] [57] **C2** [105] [106] ***Performance Boundary:*** **C1** [65] [67] **C2** [66] ***Hazardous Event (ISO 26262):*** **C4** [32]	***KPIs in Simulation:*** **C1** [107] [108] [89] [109] [110] [111] ***Driveable Area:*** **C1** [69] [70] ***Collision:*** **C2** [94] [112] **C3** [113] [54] [56] [76] [112] **C4** [61] [30] [62] [63] [32] [114]
Function-Critical	Consequence Aware: ***KPIs in simulation:*** **C1** [78] **C2** [47] ***Formal Specification:*** **C1** [57] **C2** [105] ***Failures:*** **C3** [104] **C5** [68] [115] [116] [117] Consequence Unaware: ***Performance Boundary:*** **C1** [80] ***Differential Behaviors:*** **C5** [118] [119]	Consequence Unaware: ***Complexity:*** **C1** [120] [121] [122] [123] **C5** [68] [124] [125] [126] ***Predictability:*** **C5** [127]
Other	***Fuel consumption:*** **C1** [80] ***Comfort:*** **C1** [81] **C2** [71] (KPI) ***Overall Traffic Quality:*** **C1** [128]	

TABLE 7
Number of studies classified according to their inputs and outputs

Out. / In.	Functional	Logical	Concrete	Criticality
Others	2	2	0	0
Functional	2	0	4	0
Logical	0	2	52	6
Concrete	3	0	12	6

definitions of criticality. TABLE 6 explicitly classifies the definitions of criticality according to the aforementioned three dimensions. As shown in TABLE 6, most of the primary studies focus on the identification of safety-critical scenarios. The number of primary studies that find implementation-specific critical scenarios is larger than those finding non-implementation-specific critical scenarios.

Different criticality definition entails different surrogate measures, which are further summarized in TABLE 6 and discussed in the next section.

5.4 Level of Abstraction

Critical scenarios can be identified on different levels of abstraction. As introduced in Section 4.1, this section classifies the CSI methods according to their inputs and outputs in terms of the level of abstraction of scenario description. The classification result is given in Table 7.

Most of the studied CSI methods find concrete critical scenarios within a given logical scenario (i.e., *logical → concrete*). An important step within this process is to evaluate the criticality of a given concrete scenario (i.e. *concrete → criticality*). These two classes cover most of the CSI methods introduced in Sections 6.1, 6.2 and 6.5.

As shown in Fig. 4, critical functional scenarios can be reasoned from the ODD definition and/or functional specifications and/or previous project experience, etc. (i.e. *others → functional*) [30], [32], [61]. In some papers, these reasoned functional scenarios are also formulated into logical scenarios (i.e. *others → logical*) [62], [63]. These methods are analyzed in Section 6.4.

Critical functional scenarios can also be induced from accident databases. Depending on whether the accidents in the database are described qualitatively or quantitatively, these methods can be classified as *functional → functional* [113] or *concrete → functional* [54]. These methods are introduced in Section 6.3. Another type of *functional → functional* method is to combine triggering conditions with hazardous operational situations [32].

Studies evaluating the criticality of a logical scenario (i.e. *logical → criticality*) actually evaluate the failure rate of an SOI under the given logical scenario. These studies estimate the failure rate through importance sampling [42], [83], [84], [85], [86], [88], which is a variant of Monte-Carlo simulation, considering the distribution of critical scenarios in the space of the given logical scenario. It should be noticed that the focus of this survey does not include how the failure rate is evaluated. We only care about how the critical region can be discovered at the first step of the importance sampling.

There are two types of *concrete → concrete* methods. The first type refines a concrete scenario to make it more critical by tuning its parameters [47], [57], [65], [71], [91], [118], [119], [124]. These methods can be found in Sections 6.1, 6.2 and 6.5. The second type requires multiple critical concrete scenarios to synthesize new critical scenarios [75], [76], [97], [112]. This type of method is introduced in Section 6.3.

6 The Solution Category

According to the similarities on the problem formulations and solutions, studied CSI approaches are grouped into the following five clusters. Fig. 14 illustrates the number of primary studies in each cluster.

C1: Exploring logical scenarios without parameter trajectories (Section 6.1)
C2: Exploring logical scenarios with parameter trajectories (Section 6.2)
C3: Inductive reasoning methods (Section 6.3)
C4: Deductive reasoning methods (Section 6.4)
C5: Finding critical scenes for CV-based functions (Section 6.5)

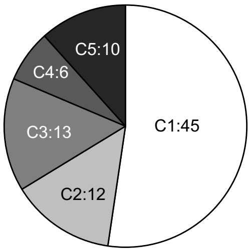

Fig. 14. Number of primary studies in each cluster

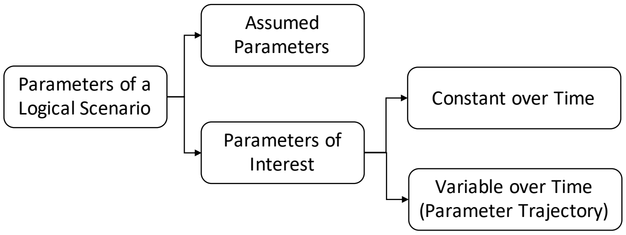
Parameters of a Logical Scenario
Assumed Parameters
Parameters of Interest
Constant over Time
Variable over Time (Parameter Trajectory)

Fig. 15. Classification of parameters of a logical scenario

As depicted in Fig. 4, a logical scenario can be instantiated into multiple concrete scenarios. The parameters of a logical scenario can be classified as illustrated in Fig. 15. Assumed parameters have fixed values for all the instances (i.e. concrete scenarios), e.g. the number of vehicles in a scenario and the number of lanes. Parameters of interest construct the scenario space to be explored. These parameters include the ones that are constant over time (e.g. the weather condition or the position of a stationary obstacle on the road) and the ones that are variable over time (e.g. the speed of a surrounding vehicle or the perception error). If a parameter is variable over time, it can be represented as a parameter trajectory. Values of the parameters can be either categorical (e.g. weather, color and vehicle model) or numerical. Numerical values can be either continuous (e.g. speed, heading and sensor noise) or discrete (e.g. the number of other vehicles, the number of lanes and speed limit).

The approaches to explore a logical scenario are elaborated in both **C1** and **C2**. The difference between these two clusters is that the logical scenarios in **C1** do not contain parameter trajectories. **C3** analyzes the methods to induce critical scenarios from different data sets. **C4** discusses systematic approaches to deduce critical functional scenarios. Most of the computer-vision (CV) based functions (e.g. object detection or classification) take a scene (i.e. a camera image) as input at each time step. The performance of such a function is mainly affected by the input scene rather than how the scene develops over time. Methods to find critical scenes for CV-based functions are summarized in **C5**.

The following subsections respectively introduce our analysis of these clusters. Except in **C4**, CSI approaches in each cluster are summarized with a gated tree, i.e. Fig. 17, 22, 23 and 24. Each node of these trees represents a component of a CSI solution. A parent node is connected with its children through either an ”AND” gate or an ”OR” gate. The ”AND” gate implies that the parent note is constructed with all its children as necessary components.

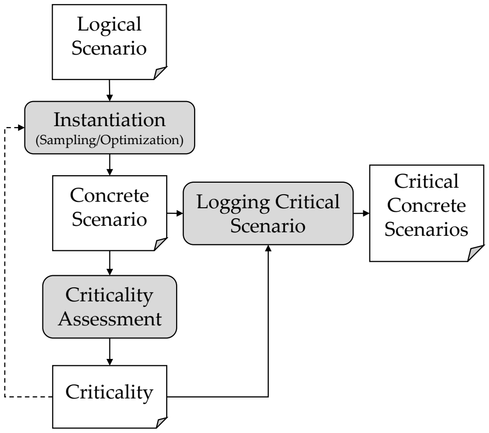

Fig. 16. Critical concrete scenario identification process

For example in Fig. 17, all the methods in C1 (i.e. the root node) contain three components, namely a logical scenario model, an exploration method and a criticality assessment approach. If they are connected with an ”OR” gate, the children will be the alternative solutions of the parent node. In other words, in a particular paper (e.g. [41]), the parent node (e.g. the Exploration Method node in Fig. 17) will choose one of its children (e.g. Combinatorial Testing) as a solution. Not so many papers have been found in **C4**. Such a systematic figure is not given for **C4**, since a conclusion drawn from such few samples will not be representative. A systematic classification of CSI methods in **C4** will be part of the future work.

6.1 Exploring Logical Scenarios Without Parameter Trajectory

This section focuses on approaches to explore a logical scenario that does not contain a parameter trajectory. The exploration methods for logical scenarios with parameter trajectories are analyzed in Section 6.2. The exploration processes in this cluster are formulated as Design Space Exploration (DSE) or Search-based Testing (SBT) problems [92], whose flow chart is shown in Fig. 16. Given a logical scenario, a set of concrete scenarios are generated with a parameter space exploration method. Among these generated concrete scenarios, critical ones are identified with a pre-defined criticality assessment method. Criticality assessment can be seen as a function that maps a point in the scenario space to a point in the scoring space. The scoring space is the quantitative evaluation of criticality for concrete scenarios. The evaluation is achieved through the surrogate measures since the criticality can hardly be measured directly. If the criticality is defined on multiple dimensions, each dimension in the scoring space refers to a criticality measure. Based on Fig. 16 and the taxonomy in Fig. 10, all the CSI methods in this cluster are summarized in Fig. 17. The analysis of Fig. 17 follows the taxonomy of *Solution* category in Fig. 10.

Regarding the analysis in Table 7, most of the approaches in this cluster take logical scenarios as inputs and generate a set of critical concrete scenarios, which are further used to construct test cases or simulation cases. Exceptions include the approaches that focus on the criticality assessment of a given concrete scenario [95], [110], and the refinement of a logical scenario [69], [70]. Criticality assessment is considered in this cluster because it is one of the essential steps to search for critical scenarios, as shown in Fig. 16. If a primary study only focuses on a criticality assessment approach, it usually implies that the given concrete scenario is generated from its previous step. Logical scenario refinement methods are included in this cluster only if they employ search-based methods. Instead of finding individual critical scenarios in the scenario space, they optimize the lower and upper bounds of the parameters of interest to derive the critical region. It can be treated as a prepossessing step of CSI methods to reduce the searching space.

The rest of this section explains Fig. 17 based on all the primary studies in this cluster.

6.1.1 Scenario Model

In this cluster, a logical scenario is modeled as a scenario space. Each dimension is a parameter of interest whose value can be either categorical or continuous. A logical scenario also specifies their value ranges or distributions. A concrete scenario is a vector in this scenario space with a fixed value for each dimension. As mentioned in Fig. 3, each parameter of interest correlates to one known scenario factor at one layer, as described in TABLE 2. From the perspective of test (or simulation) case construction, each parameter specifies either:

- an **initial condition** including, e.g., the initial position, speed, or heading of the ego vehicle or other traffic participant;
- a **constant condition** including the weather condition (e.g., sunny, foggy, or raining) and the road topology;
- or a **parameter of an assumed model** including, e.g., the size, field of view, or constant speed of a vehicle model

Parameter distributions can help to estimate the likelihood of exposure of a given concrete scenario. 9 out of 45 studies in this cluster consider realistic distributions of parameters ([42], [45], [82], [83], [84], [85], [86], [88], [95]) during the exploration. Other studies assume that the parameters follow a uniform distribution. The realistic parameter distributions are taken into account mainly for the following reasons:

- **To estimate the failure rate of the SoI in all situations:** As discussed in Section 5.4, estimating the failure rate with importance sampling needs critical scenarios or critical region as input. In addition, it also needs to know the likelihood of exposure of each sampled concrete scenario, which can be approximated based on the parameter distributions.
- **To consider commonality as part of the criticality definition:** Compared to rare cases, those with higher probabilities to exposure in the real world are of significant interest. To this end, the commonality of a scenario is introduced into the definition of criticality [42], [82], [95], so that the exploration method can find common and hazardous scenarios. This commonality can be quantified with the help of parameter distributions in real traffic.

Parameter distributions are derived from an existing real-life driving database (i.e., the Naturalistic Driving Study (NDS) or Field Operational Test (FOT) data). The parameters can be assumed either mutually independent [110], [111] or dependent [82]. Akagi et al. [82] considered the parameter distribution and approximate it by a Gaussian Mixture Model (GMM), where parameter probability distributions represent the covariance of variables. Methods to approximate parameter distributions from a given data set are summarized in [129].

6.1.2 Exploration Methods

Fig. 17 lists all the methods to explore the scenario space employed in the primary studies in this cluster. These exploration methods can be divided into two types: (1) naive search (i.e., *Sampling* and *Combinatorial Testing*) and (2) guided search (i.e., *Optimization* and *Learning-based Testing*).

A naive approach for scenario exploration is to search randomly or systematically over the scenario space. In other words, samples are mutually independent. Therefore these approaches can be implemented in a parallel manner to reduce the exploration time. However, if critical scenarios are rare, these approaches can be inefficient as the probability of sampling a critical scenario is low. On the other hand, the guided search methods have the potential to be more efficient, since the searching direction at each iteration is adjusted according to the search result of the previous iteration, so as to converge the exploration to critical regions. Each exploration method under these two types is briefly introduced in the rest of this section.

Sampling: The sampling method instantiates a concrete scenario by randomly assigning each parameter’s value in a logical scenario space. A predetermined number of samples are taken statistically based on probability distributions of parameters, and its sampling size is determined by the required coverage and computation time for simulation. The applied sampling methods are summarized in Fig. 18. According to the parameter descriptions in logical scenarios, the sampling approach can be classified based on whether the parameter distributions are taken into account. We assume that a uniform distribution is adopted if the parameter distribution is not mentioned. Near-random sampling, such as Latin Hypercube sampling [67], can improve the coverage when the sampling size is small. It splits the multidimensional parameter space into even grids and selects samples in each grid with a given number.

As mentioned in Section 6.1.1, parameter distribution is considered for two purposes: failure rate estimation and commonality consideration. Since the failure rate estimation by importance sampling barely appears during the scenario space exploration phase, in this section we only consider parameter distribution used to model the commonality of the scenarios. Moreover, when considering the parameter distributions, relationships between parameters can be assumed either dependent or independent. Different assumptions require different sampling methods. Studies

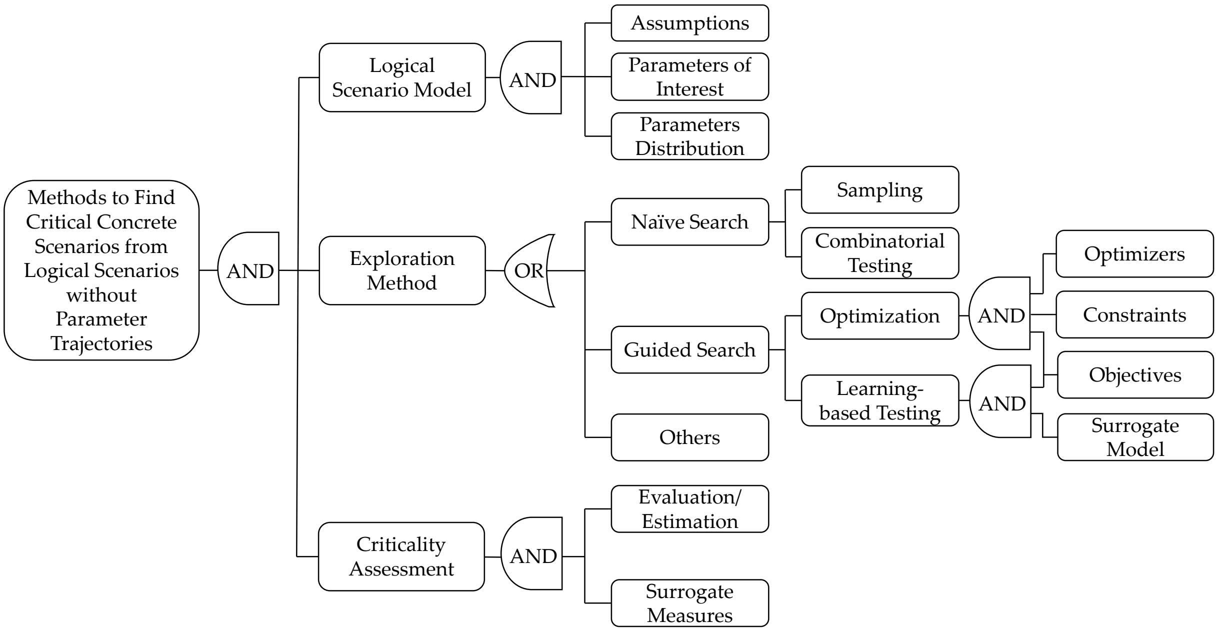

Fig. 17. Instantiation methods without parameter trajectories

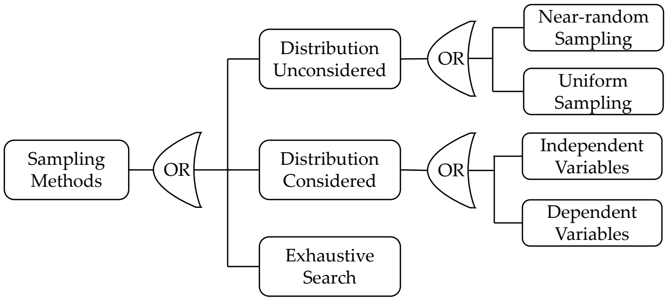

Fig. 18. Sampling method in scenario exploration

[110], [111] investigated the parameter distributions before sampling. In these studies, each dimension in sampled scenarios is viewed as independent and identically distributed (i.i.d.). The covariance among different parameters is not considered, and their values can be determined independently. Nevertheless, correlations among variables are also investigated in some other studies [82], [107]. When considering the relationship among variables, more restrictions are imposed on the sampling. Instead of Monte-Carlo sampling, the Markov Chain Monte-Carlo (MCMC) method [82] can be applied efficiently with the knowledge of parameter distribution and covariance. The proposed approach in [82] applies a risk index from naturalistic driving data to select risky traffic conditions efficiently.

If the scenario space is fully discrete and reasonably small, an exhaustive search is applicable to guarantee full coverage. As shown in Fig. 18, it is considered as an extreme case of sampling. If the scenario database is available, the exhaustive search can be achieved through checking every test case derived from NDS/FOT data [53]. Otherwise, the exhaustive search can discretize continuous parameters and perform a full-scale grid search to examine every existing test case in the scenario space [43], [89], [107].

Combinatorial testing: Combinatorial Testing is a commonly used software testing method focusing on the identification of failures triggered only by particular combinations of inputs [130]. The core of the combinatorial testing is to generate a minimum set of test cases (i.e. a covering array) that satisfy N-wise coverage[11]. Under the context of CSI for ADS, combinatorial testing can be used to find the unknown combinations of influential factors that may fail the ADS or a particular AD function. An example is when a pedestrian crosses a road in front of the ego vehicle in the evening. The criticality is significantly influenced by the combination of different scenario parameters, such as the initial speed of the ego vehicle, visibility, and distance between the pedestrian and the vehicle. The covering array, as a matrix that stores the testing configurations, specifies test data where each row of the array can be regarded as a set of parameter values for a specific test [132]. For example, if the required coverage is three-wise, the generated covering array should cover all combinations of values from arbitrary three parameters, where three specifies the number of parameters in combination. A good covering array can significantly reduce computational cost and improve test efficiency. However, finding the minimum covering array is an NP-hard problem [130]. A test database comprises all test scenarios in the covering array, and it can cover all possible combinations of parameter values at a predefined degree.

N-wise coverage can only be defined on discrete parameter space. However, in a real-life scenario, both continuous and discrete variables are prevalent in the parameter space of the logical scenario. A common method to handle continuous parameters is discretization, which is a process of quantizing continuous attributes by converting the continuous data into a finite set of intervals and assigning some

[11] N-wise coverage states that all N-tuples of parameter values must be tested at least once, given an SoI model with the parameter list, values, and constraints to define parameter interactions [131]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

specific data values to each interval [133]. In some other studies, Tuncali et al. [65] circumvented the problem by a two-step approach. The first step performs combinatorial testing on all the discrete parameters. In the second step, for each combination of the discrete parameters, they conducted optimization by simulated annealing on all the continuous parameters. The optimization is used as the falsification process to identify critical scenarios.

Instead of focusing on the minimum number of test cases in the covering array, covering array generation methods can also be customized. For example, in [120], [121], [122], [123], the generated covering array fulfills not only the N-wise coverage but also maximizes the overall complexity of all the scenarios.

Optimization: Studies in this class formulate the CSI approaches into optimization problems, which generically contain four parts, namely the design variables, the constraints, the objective functions, and the optimizer (i.e. the solver) [134]. The design variables have already been analyzed in Section 6.1.1 and their ranges are restricted by the constraints. Therefore, as shown in Fig. 17, this section mainly analyzes the objective functions and the selected optimizers.

In the vast majority of studies in this class, the objective functions represent quantified criticality measures, which are discussed in Section 6.1.3 in detail. Other than criticality, the objective function can also include the similarity between scenarios (i.e. the distance in the scenario configuration space) to maximize the diversity of the identified critical scenarios [83], [84].

The choice of optimizer highly depends on the problem formulation, especially the transparency and complexity of the underlying models. A simulator is commonly employed when estimating criticality measures. Due to the high model complexity and unavailability of the interior structure of plant models, in many cases simulation and system analysis of the SoI can be computationally expensive. For this reason, the behavior of the SoI is treated as a black box. Thus, the corresponding search process can be regarded as a black-box optimization problem. For a black-box optimization problem, the input (i.e. scenario parameters) and output (i.e. simulation results) relationship can only be analyzed by exterior observation through simulations [135]. To tackle this problem, various heuristic methods can be applied to approximate the fitting function and find the global minimum iteratively, including genetic algorithm [49], [50], [108], Bayesian optimization [92], [98], and simulated annealing [51]. In addition, the domain-specific heuristic technique, where the optimizer only suffices in a certain system, helps to identify multiple local minimums of feasible solutions by rule-based searching [111], heuristic simulation-based gradient descent [44] and zoom-in sampler [52]. Scenario searching is formulated as a two-step optimization problem in [83], [84], [85]. The first step of the optimization tries to find multiple local optimal solutions. In the second step, the neighborhood of the local optimal solutions is searched to find all the scenarios whose criticalities are within a given threshold.

Learning-based testing: Learning-based testing methods [136] aim to automatically generate a large number of high-quality test cases by combining a model checking algorithm with an efficient model inference algorithm, and integrating these two with the SoI in an iterative loop. It learns the properties of the SoI during optimization by training a surrogate model. The diversity can be optimized by maximizing the distances between samples in either the scenario space or the scoring space. Mullins et al. [80] used a surrogate model for training to improve the sampling coverage. It takes a set of samples as input and returns the estimated diversity (i.e. the mean distance) on the scoring space of the input samples. Nabhan et al. [81] used learning-based methods to try to maximize the distance between scenarios. It also considers non-safety-critical qualities (such as deceleration and jerk effects indicating passenger's comfort) during the scenario generation.

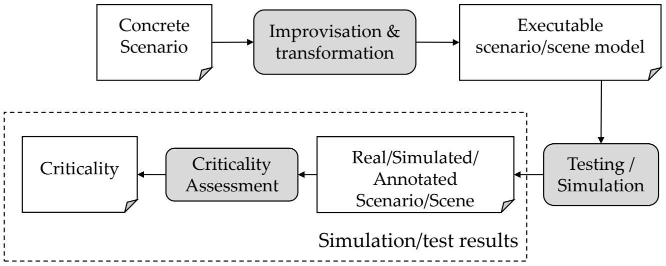
Concrete Scenario
Improvisation & transformation
Executable scenario/scene model
Testing / Simulation
Real/Simulated/ Annotated Scenario/Scene
Criticality Assessment
Criticality
Simulation/test results

Fig. 19. Test case generation from concrete scenarios and testing-based criticality assessment

Others: This category refers to other methods to generate test cases and detect critical scenarios not included in our research, such as Tabu search, hill-climbing, and grammatical evolution. The review of common search algorithms in software testing can be found in [137], [138].

6.1.3 Criticality Assessment Methods

After attaining concrete scenarios from the CSI methods mentioned above, most of the primary studies adopt testing-based approaches to verify the criticality of derived scenarios, as shown in Fig. 19. Different definitions of criticality are discussed in Section 5.3. This section discusses how to assess the criticality of a concrete scenario under different criticality definitions.

In the criticality assessment phase, most of the studies utilize an X-in-the-loop simulation to estimate the criticality of a concrete scenario, where X represents the model, software, or hardware of the SoI as a black-box. Criticalities are assessed on the simulation results based on the predefined surrogate measures summarized in Table 6. Nevertheless, the criticality can also be assessed without X-in-the-loop simulations. Validation can be realized through real-world testing to analyze the performance of the proposed exploration method [84], [89].

As discussed before, criticality assessment is a function whose input is a concrete scenario, and whose output is a quantified criticality index. Most of the studied CSI methods in this cluster implement this function deductively with analytical approaches, while the others implement it inductively with machine learning approaches [80], [81].

Inductive criticality assessment is generally applied in the data-driven approach. Instead of evaluating scenario criticality by the black-box simulation, it constructs an approximation model to emulate the real system behavior with the database. The database is composed of existing scenario data with labels, which indicates the criticality of the scenarios and is used in the training process of the approximation models. The labels of criticality measures can be attained directly from the training data in the scenario database. It can also be derived by simulation of concrete scenarios, where the simulation result is regarded as the ground truth.

Since criticality can hardly be measured directly, deductive criticality assessment is based on a surrogate measure to evaluate the criticality of a concrete scenario quantitatively. The surrogate measures can be either Boolean or numerical in the scoring space, depending on the types of exploration methods. For naive search methods, criticality measures are Boolean, since each sampled scenario needs to be evaluated as critical or non-critical. For guided search methods, since criticality is a part of the objective function, the criticality measure has to be quantified as numerical. Under each criticality definition, one or more surrogate measures were proposed in different primary studies, as summarized in Table 6. More measures of criticality can be found in [139]. Based on Table 6, the surrogate criticality measures used in this cluster are summarized as follows.

KPI-based measures are commonly used in this cluster due to the simplicity of implementation. It describes the proximity to a critical state through simulations driven by black-box testing. KPI-based methods calculate metrics by assessing a posterior measurement of the vehicle state. The metrics are used to evaluate the criticality in a scene, and the criticality of a scenario can be subsequently indicated by the worst scene. In the context of the safety-critical scenario measurement, KPI metrics refer to Time-to-X metrics (e.g., Time-to-Collision, Time-to-Brake, and Time-to-React), distance-based metrics (e.g., longitudinal and lateral distance), velocity-based metrics (e.g., relative speed), and acceleration-based metrics (e.g., required deceleration). A comprehensive analysis of KPI comparison can be found in [140]. The metrics above are usually used as the measures of a particular function. However, compared to the implementation-specific category, the non-implementation-specific ones can be found only with the generic features of the SoI, for example, a highly abstract simulation model which is generic enough to represent different implementations of the SoI or a criticality measure that is only based on the environmental aspects. The critical scenarios are detected without information of the implemented function, which makes it possible to apply scenarios for system design [110] and early phase of verification [89]. Also, **the driveable area** [69], [70] can be viewed as a special measure in this category, where the criticality is defined directly on a logical scenario by examining the range of relevant parameters. The SoI (e.g., general motion planning algorithms) will be more critical if the solution space is smaller with a less drivable area. Meanwhile, besides safety-critical applications, KPI-based approaches can also be applied in functional-critical studies. Unlike safety-critical measures, function-critical assessment emphasizes the performance of a particular module of the SoI (e.g., sensor, decision-making, and actuator), which does not necessarily propagate to an accident.

Besides the KPIs mentioned above, in order to increase the efficiency of critical scenario detection, **complexity** can be regarded as an auxiliary property of criticality in some studies [120], [121], [122], [123]. Compared to traditional software testing, critical scenario generation focuses on finding triggering conditions instead of explicit software bugs. Different value selections correspond to different levels of complexity, which is treated as a priori knowledge before performing the test case generation. By our definition in Section 5.3, they are viewed as non-implementation-specific and consequence-unaware, since the complexity is not exclusively oriented to any particular malfunction type.

Compared to KPI-based methods, **collision** is a more straightforward measure in the safety-critical assessment. It generates binary output according to whether a crash happens in a simulated scenario. However, some real-world collision scenarios may not manifest on simulation due to the low fidelity of the simulation model. In this cluster, collision analysis only appears in the implementation-specific category since collision avoidance has to be realized with certain types of ADAS/ADS.

A more complex and specific measure in implementation-specific class can be represented by **formal specifications** such as signal temporal logic (STL) [57], [65]. This kind of method can be applied to either safety-critical or function-critical use cases. The safety-critical approach examines the scenarios that may lead to accidents, while the function-critical measure aims to find anomalies in subsystem-level function and test its robustness.

In addition, implementation-specific criticality can also be characterized by **performance boundary**, which can be divided into two types. The first type is the predefined boundary, where the exploration methods try to separate critical scenarios to non-critical ones through the boundary and find avoidable collisions [67]. In the second type, the performance boundary is unknown at the beginning. Several performance modes are derived through clustering the scoring space. Scenarios around the boundaries of the performance modes are of great interest since slight changes of parameter values can contribute to the behavior change. It is assumed that faults tend to manifest in those scenarios [65]. Moreover, we distinguish the study in [80] as the consequence-unaware type for the function-critical use case, since a scenario is regarded as critical if a small change of its configuration leads to significant changes in the SoI performance. By this definition, consequential malfunctioning behavior is not explicitly given. In the above-mentioned types of studies, the criticality assessment is realized by simulation.

Apart from the safety-critical and function-critical consideration, **quality of service (QoS)** indices can also be applied as surrogate measures. In this cluster, the normal operations of the vehicle are assumed, and QoS is quantified to judge the performance of a system, such as by fuel consumption [80], passengers' comfort [81], and overall traffic quality [128].

6.1.4 Coverage

For safety argumentation, coverage must be considered when exploring a logical scenario. However, not all the primary studies in this cluster explicitly discuss coverage. As

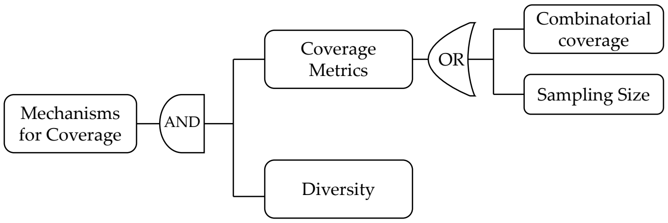

Fig. 20. Mechanisms for coverage of scenario space exploration methods

shown in Fig. 20, this section summarizes the consideration of coverage and the mechanisms to increase the diversity of the identified critical scenarios in this cluster.

Combinatorial coverage (i.e. N-wise coverage) [41], [57], [65], [120], [141] and sampling size [67], [89], [110] are the two metrics found in the primary studies in this cluster to measure the coverage of the exploration. These two coverage metrics are detailed in Section 6.1.2.

Another definition related to coverage is the diversity of the identified critical scenarios. Similar scenarios have a high potential to reflect the same triggering condition. Diverse critical scenarios can help to identify different functional insufficiencies and hazardous events. The rest of this section discusses mechanisms to increase the diversity of the identified critical scenarios.

In the primary studies, diversity is quantitatively defined based on the distance in the scenario space. For sampling-based exploration methods, distances are generally estimated in the Euclidean space. Different measures, such as Voronoi interpolation [92], can be used at sampling to ensure no tests exist in close proximity. Moreover, various sampling methods, such as Latin Hypercube Sampling [67], can also improve the diversity of samples. It divides each parameter into intervals and ensures the scenario space is evenly covered.

On the other hand, for optimization or learning-based methods, diversity can be considered part of either the objective or the constraint. The diversity can be realized either by examining the distance among scenarios [80], or by exploring multiple local minima [91], [98]. For the latter approach, Wagner et al. [91] extracted key features from critical scenarios by principal component analysis (PCA) and added noise in a lower-dimensional component space to reconstruct more varied critical scenarios. Gangopadhyay et al. [98] applied Bayesian Optimization to identify multiple minima regions from a non-convex function, where critical scenarios are situated in minima regions. Both methods mentioned above aim to find critical scenarios with diverse failure conditions.

6.1.5 *Required Information*

When identifying critical concrete scenarios from a logical scenario, a CSI method needs a logical scenario as input and one or more surrogate measures to support criticality assessment. These have been discussed respectively in Sections 6.1.1 and 6.1.3. Research in this cluster normally verifies the case study by simulation-based approaches. In the context of simulation, the completeness of a test case includes four parts, namely simulation objects (i.e. SoI), simulator and environment, criticality, and database. The simulation objects and criticality have been discussed in previous sections. Herein, we mainly focus on the simulator and database.

Simulator: The simulator provides the platform to implement and express SoI properties in vehicle dynamics and ADAS/AD function through modeling language. Many high-fidelity simulators (e.g., PreScan, CarMaker) also support virtual test driving to visualize simulation environments. We regard a simulator as the ad-hoc simulation platform if the simulator is developed in-house or not mentioned in the article. MATLAB, together with other programming language simulators, are viewed as low-fidelity simulation platforms as the system dynamics are simplified, or virtualization is omitted. The main target of the low-fidelity simulator is the realization of a system dynamics description and to enable the process of sampling, thus the model fidelity is not the focus.

Database: All primary studies referring to the database in this cluster employ it to analyze parameter distribution as prior knowledge. Based on our findings, we cluster the database into three types: naturalistic driving data [89], [90], [91], [109], traffic flow data [53], [82] and collision accident data [107]. It is worth noting that NDS databases exclusive to autonomous driving (e.g., Safety Pool) are available but have not been found applied in the field of this cluster.

6.2 Exploring Logical Scenarios with Parameter Trajectories

In the previous section, the behavior of other actors (e.g., vehicles or pedestrians) is determined by several parameters of a predefined motion model (e.g., the velocity of a constant velocity model). In this way, possible behaviors of other actors are restricted, especially their reactions to the behaviors of the ego vehicle. A more arbitrary way to determine the behavior of another actor is to model it as a motion trajectory (position trajectory or acceleration trajectory). This can also be extended to any parameters whose development over time is of interest (e.g., the angle of the sun). In general, the variable over time parameters, also called parameter trajectories, are the values of the scenario that change during the simulation. Methods to explore a logical scenario including such parameter trajectories are discussed in this section. Such problems are also called stress testing in [58], [60], [96] or adversarial testing in [105].

Replacing one parameter with a trajectory of parameters will significantly increase the exploration space. Therefore, the methods introduced in the previous section are not fully suitable for the problem in this cluster. The rest of this section discusses the methods in the primary studies for this type of problem.

In these studies, the SoI is challenged by a scenario in which the behavior (or the perceived behavior) of the other actors is set to make the system fail under a certain metric. Some methods treat the SoI as a back box e.g. [69], [70] while others may need to know the model of the system as part of the optimization process e.g. [71], or at least the order of the model [47]. Fig. 21, based on [142], shows the relationship between the SoI, the environment and the adversarial agents. The SoI generates actions based on the

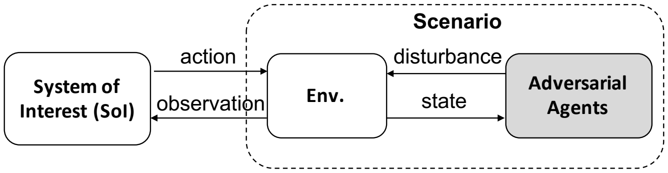

Fig. 21. Relation between the SoI, the environment and the adversarial agents [142]

observations obtained from the environment, which is also influenced by the adversarial agents. All the connections represented in this diagram, the actions and observations from the SoI, and the states and disturbances from the adversarial agents, are parameter trajectories.

An overview of methods found in this cluster can be seen in Fig. 22. These methods require a scenario model, an exploration method, and a criticality assessment based on the taxonomy in Fig. 10. The scenario model contains information about the constraints imposed to the scenario. The exploration method searches for a scenario that may lead the SoI to a hazardous event (as described in Fig. 3) based on the criticality assessment. This criticality is assessed based on a pre-defined surrogate measure.

The categories proposed in [142] have been used to classify the exploration methods of this section. Even though the scope of [142] is not within the automotive domain, and the methods analyzed only consider black-box safety validation, the proposed categories are still useful for the purpose of classifying the exploration methods of the present study.

6.2.1 Scenario model

A scenario model includes a set of parameters. Some of them have a predefined value (assumed parameters) while others (parameters of interest) should be optimized to find a critical concrete scenario, as shown in Fig. 22. Examples of assumed parameters in the primary studies are the number of other actors (moving traffic participants) [94], the number of lanes [69], and the position of the crosswalk [58].

Referring to Fig. 15, parameters of interest in this cluster may include both constant (over time) parameters and parameter trajectories, or only parameter trajectories. An example of constant parameters is the the number of agents in a particular scenario in [94], which is explored within a predefined range.

The exploration method aim to find the optimal parameter trajectories to challenge the SoI. These parameters generally include the motion profiles of other actors and the level of noise (e.g., [58]). Parameter trajectories can also be used to model other factors such as weather conditions or traffic light patterns, as mentioned in [105].

Motion profiles can be restricted to longitudinal motion, as in [46], or they can also include lateral positions. The lateral positions can be expressed with respect to the center of the lane, as in [106] or using absolute coordinates, as in [58]. Besides positional restrictions, there can be other types of constraints, as in [106], where the lateral velocity and the yaw rate were specified to be within certain values.

Including parameter trajectories significantly increases the searching space. In addition, the values of the same parameter at different time steps are not independent. Therefore, methods introduced in Section 6.1 are not optimal for the problems in this cluster. Instead, most of the approaches in this cluster formulate the exploration as a sequential decision-making problem.

6.2.2 Exploration method

The exploration method aims to find a set of values and trajectories for the parameters of interest that force the SoI to fail under a certain metric. As mentioned before, for the classification of the methods, the categories proposed in [142] have been followed, where these exploration algorithms are divided into optimization, path planning, and reinforcement learning:

Optimization: These methods are similar to the ones introduced in the previous section. They assume that the values at different time steps of a trajectory are independent and include a cost function designed to guide the search for a trajectory that forces the SoI to fail. In [142], simulated annealing, genetic algorithms and Bayesian optimization are included in this category. [100] is the only publication related to this category that has been identified. It uses Bayesian optimization to generate adversarial scenarios. Bayesian optimization is an algorithm proposed to optimize functions that are expensive to evaluate without knowledge about the structure of the function e.g. concavity or linearity [143]. In [100], the optimization is not directly used to generate a trajectory. It is used to optimize the policy of the adversarial agents that decided their behavior based on their observations. During our literature search, no primary study has been found that uses optimization techniques to directly generate trajectories. One of the reasons might be the complexity of the environment, which might make these methods impractical due to the exponential growth of the state space.

Path planning: Such algorithms aim to navigate the state space from a starting state to a goal state. In this case, the goal is to get from an initial state to one of the failure states using disturbances as control inputs [142]. In this category, the following algorithms have been identified:

Rapidly exploring Random Trees (RRT): RRT is a search algorithm for path planning designed to handle high degrees of freedom [144]. This method is used in [46] to make an ACC system fail by generating motions of the leading vehicle. They use two methods to explore the tree: forward search and backward search. Forward search starts at a randomly generated safe state and generates the tree based on the actions of the leading vehicle aim to get the ego vehicle into an unsafe state. Backward search starts at a random unsafe state and searches backwards in time trying to reach a safe state. In both methods, they randomly generate the behavior of the leading vehicle and, based on that, they generate the response of the ego vehicle. In [66], the aim is to find almost-avoidable collisions or near-misses. To promote this kind of collision, RRT was used together with a custom cost function that estimated how avoidable a collision was based on the ratio of the collision surface, the collision speed, and the minimum time to collision. In both studies, each node of the tree includes the state of the system so that, when the tree grows, only a partial simulation is executed from an existing node of the tree to the new node, instead of running the simulation from the initial state.

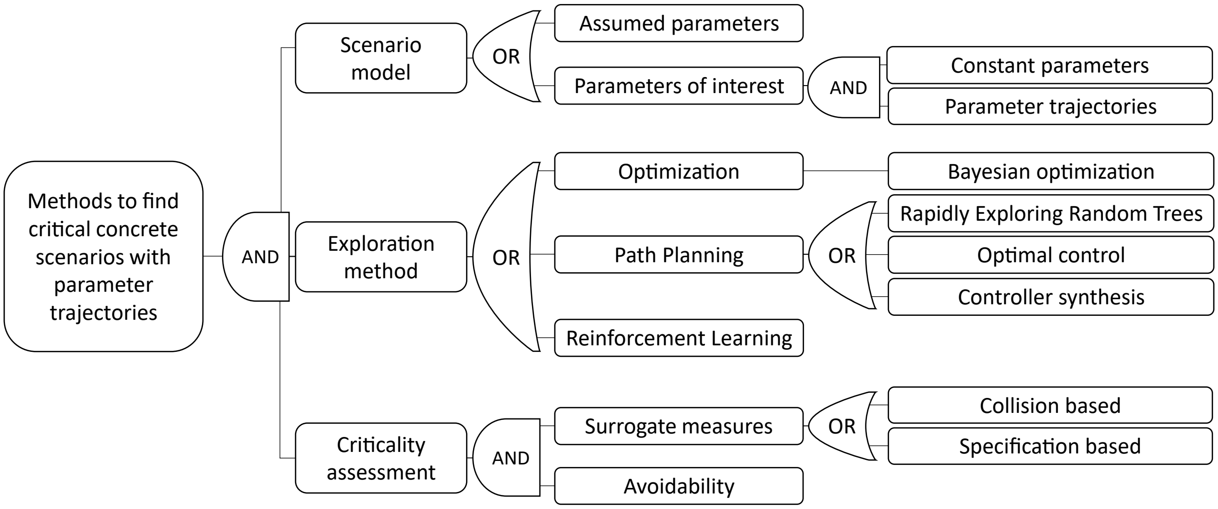

Fig. 22. Instantiation methods with parameter trajectories

Optimal control: This is a control strategy that aims to optimize an objective function. It originated as an extension of the calculus of variations [145]. In [47], the process to search for critical scenarios is formulated as an optimal control problem, which finds the optimal sequence of control input (in the case study, it is the acceleration profile of the leading vehicle and the time instances of critical scenes) to optimize the objective function.

Controller synthesis: It aims to generate of a controller given the complete model of a system and the environment that is guaranteed to achieve a specification [146]. In [106], the authors use controller synthesis to generate scenarios where it is guaranteed to be possible to satisfy a set of formally defined specifications.

Reinforcement learning: It aims to program agents by giving a reward based on their actions without specifying how the task should be accomplished [147]. Actions might affect not only the current reward, but also the future ones [148]. The algorithms analyzed in this section model the problem as a Markov decision process (MDP). The methods found in this category are the following:

Monte-Carlo tree search: It is a heuristic search algorithm for decision making that takes random samples in the decision space and uses that information to build a search tree [149]. In [58], the authors compare its performance against deep-reinforcement learning and Trust Region Policy Optimization (TRPO) to solve the MDP used to model the scenario.

Deep reinforcement learning: Such algorithms use deep neural networks to approximate the value function, the policy or the model of reinforcement learning [150]. Koren et al. [58] extended Adaptive Stress Testing (AST) originally proposed in [151] to test aircraft collision avoidance systems. AST formulates the scenario as an MDP and uses a Monte-Carlo tree search to solve it. In [58], the authors proposed to extend AST by using deep reinforcement learning to solve the MDP and apply it to a set of autonomous vehicle scenarios. The studies in [59] and [96] improve the reward function previously proposed by using RSS (Responsibility Sensitive Safety) [7] to find scenarios where the ego vehicle performs improper actions and including a trajectory dissimilarity reward to find diverse failures. In [60], the authors extend the work performed in [58] by replacing the original Multi-layer Perceptron Network with a Recurrent Neural Network. In [105], the authors use Deep Q-learning (a deep reinforcement learning method) to learn the behavior of the adversarial agents. The ego vehicle and the agents have Rulebooks (originally proposed in [152]) associated with them to constrain their behaviors.

6.2.3 Criticality assessment

The criticality assessment is used to define whether the SoI has failed to accomplish its requirements. This assessment can be categorized based on the surrogate measure used to determine the failure and based on whether the avoidability of the failure is analyzed or not.

The surrogate measures used in the literature of this study can be categorized as follows:

Collision based: Only a collision is considered when analyzing the performance of the SoI. In [58], any collision is considered critical by the algorithm, regardless of how the collision occurs. For this reason, in some of the results obtained, the pedestrian is considered responsible after studying the scenario. In [46], the scenario model only allows for the generation of rear-end collisions caused by the ego vehicle, removing the need of having to identify the liable actor.

Specification based: In [59], the rules from RSS [7] are used to define the improper actions evaluated in the reward function. The reward function guides the search for scenarios in which the ego vehicle behaves inappropriately. In [105], the objective function is based on a Rulebook [152]. In [47], the behavior of the other actors is optimized to make the system violate its requirements.

Looking at how the avoidability of the failure is taken into account, the following classification can be made:

Avoidability not considered: Most of the studies reviewed do not consider whether the failure could have been avoided or not. They simply compute that it was a failure.

Avoidability considered: In [66] the aim is to generate test cases in the boundary where the autonomous vehicle can no longer avoid a collision (almost-avoidable collisions

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

or near-misses). To achieve that, they propose their version of RRT with a custom cost function that promotes collisions that are almost avoidable. Chou et al. [106] proposes a method to find avoidable critical scenarios using controller synthesis. Avoidability is derived with a controlled invariant set (a subset of the safe states). States in this set are such that, as long as the disturbance is within a certain range, it is always possible to find a control input so that the next state is also in this set.

A complete classification of all the papers analyzed in this study based on the surrogate measure can be found in Table 6.

6.2.4 Coverage

Coverage is not mentioned by most of the papers in this section. Some of them include a measure to promote better coverage, but none of them address it as the main focus. In [66], a novelty function is computed based on the Mahalanobis distance to achieve better coverage and avoid local minimum. The study in [59] includes a trajectory dissimilarity reward to promote the discovery of highly diverse failure scenarios.

6.2.5 Required information

All the studies analyzed in the section require a simulator. Some studies require access to the internal state of the simulator, as in [58], while others treat them as a black-box as in [60]. In [46], as mentioned before, a backward search is performed, where their method tries to get to a safe state from a future unsafe state. Due to the impossibility of computing the inverse of the ACC control law, the authors generate random inputs for the ACC vehicle to try to get the vehicle in the previous state. Then they simulate forward to ensure getting into an unsafe state again.

6.3 Inductive Reasoning

This section focuses on inductive reasoning methods. With these methods, the critical (functional or concrete) scenarios are induced from different data sources. The main data sources identified in the primary studies are summarized into two types: 1) based on accident scenarios only (Section 6.3.1), and 2) based on various types of data and scenarios (Section 6.3.2). The former source relies on the accident database, including raw accident data, accident reports or records, while the latter source refers to the existing logical or concrete scenarios, natural driving data, traffic data or sensor data. CSI methods in this cluster are summarized as shown in Fig. 23, which is also used as a basis for the structure of the following subsections (data type, reasoning methods and criticality assessment).

6.3.1 Based on accident scenarios

These methods aim to find the common features for critical scenario identification from a variety of accidental scenarios. As shown in Fig. 23, we analyzed each type from three perspectives: data type, reasoning methods and criticality assessment.

Data type: The articles [54], [55], [56], [75], [102], [103], [103], [113] used accident database (e.g., NHTSA [153], GIDAS [154], IGLAD [155] to find the critical pre-crash scenarios. The used data type of the data source can be mainly grouped into sensor data, unstructured accident records (e.g., natural language document) and structured accident records (e.g., meta-data database). The sensor data refers to the recorded time series data in accidents (such as video data from the equipped camera sensor and GPS speed, yaw rate, acceleration, and target vehicle information from equipped radar sensor [55]). The unstructured accident records are the police accident reports or protocols. In these reports, information for each scenario includes time, location, vehicle and driver details, before-crash maneuver, triggering events, crash descriptions of police officers, etc. These data are used for scenario classification and textual analysis. The structured accident records are meta data, characteristic data and categorized attribute data set from database [54], [56], [75]. These data types are mainly used for clustering analysis.

Reasoning methods: The main reasoning methods we identified are clustering, textual analysis, and reconstruction & simulation. As a typical technique for statistical data analysis, clustering is widely used in many fields and domains. In the autonomous driving domain, clustering is also used for deriving and extracting typical or relevant driving scenarios from real driving data or databases which contain a large number of scenario instances. In this section, as the first observed reasoning method, clustering was applied to find typical pre-crash scenarios from accident scenarios databases [54], [56], [156]. The derived scenarios are mostly functional scenarios, and they can be used to supplement the existing test suite or as new specific scenarios. In these papers, clustering methods need to be applied to well-defined data. Therefore the first step of these methods is to interpret source data into a pre-defined feature vector (an n-dimensional vector of numerical features that represent some scenario variables or attributes). The selection of clustering methods depends on the data types of the features. The data type of each feature can be either numerical or categorical. If the feature vector contains categorical data, clustering methods based on euclidean distance cannot be directly adopted, such as the case in [54]). Clustering methods for categorical data are introduced in [157]. Features are selected based on different analyses, such as correlation analysis or relevance analysis [56], [156](Most articles do not explain how these features are selected, more content regarding features could be found in section ontology 8.5.) The data is subsequently prepared for clustering. Typical clustering methods include k-means and k-medoids [54], [56], [156]. After clustering, scenario groups or clusters are formed based on the clustering results. One typical scenario is selected from one cluster with different methods. In the end, the derived pre-crash scenarios can be used as inspiration for the V&V testing activities. In addition, they will be used to compare against the current test suite and to supplement it.

The second observed method is textual analysis. Since accidents in accident databases are normally recorded with plain text or structured plain text, textural analysis or Natural Language Processing (NLP) techniques can help to extract features of accidental scenarios. Using NLP techniques, the accident description was parsed according to a pre-defined scenario ontology and generated typical critical

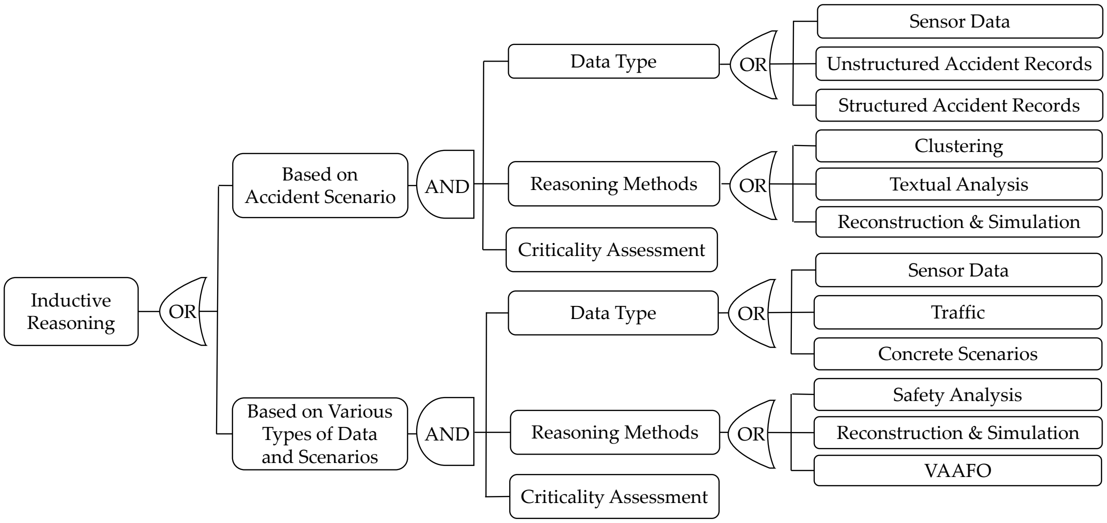

Fig. 23. Inductive reasoning methods to find critical scenarios

scenarios based on a pre-defined sentence template [102], [103], [103]. Information about the crash contributing factors (like weather, lighting, roads and the vehicles involved in the crash) were extracted from the reports. According to the parsed information, simulations were generated which can represent the accident scenarios depicted in the database. Trajectories of dynamic objects are modeled as a vector of way points. One vehicle in the simulation was then replaced with the ego-vehicle which was checked then to see if its behavior passes the oracle, which is to safely reach a goal point within a given time.

The simulation's primary role in this case is to use the simulator to ensure the safety and performance of SoI in accident scenarios. The main steps are to analyze the incident database scenarios, reconstruct them, define the oracles, and finally execute the simulation with the scenarios. Reconstruction in this case means that the data described in the database was reconstructed to the executable scenario in a simulator. Based on simulation results, critical scenarios are identified using a pre-defined set of KPIs to represent the criticality. The scenarios are derived from an accident database and the ADS is run to elicit the corresponding decisions from the system. In [55], the authors first select accident scenarios from intersection accident data according to two sensor systems (camera-based and radar-based). All the selected scenarios are classified into sixteen vehicle-to-vehicle pre-crash classes. Corresponding Safety Remaining Distances (SRDs) were calculated for each scenario for the two sensor systems. An SRD greater than zero suggests that it is possible to prevent an accident. Prevention rates for each system in each scenario class are calculated and subsequently compared with the compliance rate of each corresponding scenario to evaluate which sensor system is necessary for each scenario class.

Criticality assessment: As shown in table 6, since all the papers in this part took the data source based on accident scenarios only, the criticality definition of these articles is all related to safety-critical. The papers [54], [56], [113] use an accident database and adopt clustering methods to find pre-crash scenarios. Therefore, their criticality definitions are all non-implementation-specific. On the other hand, the studies in [55] and [75] used reconstructed scenarios and then simulated them using collision as criticality to find critical scenarios. Therefore, they are all implementation-specific. Some studies [102], [103], [103] used NLP techniques and also collision as criticality simulation. Therefore they are also implementation-specific.

6.3.2 Based on various types of data and scenarios

Compared to the previous section, articles [76], [97], [104], [112] in this section do not use accident databases as the data source to find the critical pre-crash scenarios. Their used data type can be mainly grouped into sensor data, traffic data and others.

Data type: The sensor data in this section refers to real sensor data from field operations. This data is mainly used for the virtual assessment of ADAS/AD functions to find critical scenarios. As shown in Fig. 23, traffic data was also used here as a data type, refering to the public map data and traffic flow data. This data was used for microscopic traffic simulation (i.e., each vehicle and its dynamics are modeled individually, no detailed individual sensor models and function inside) parameterization to find critical scenarios. The other data types are concrete scenarios from available analyses or simulations. These data types are mainly used for the generation of new accident scenarios from prerecorded data.

Reasoning methods: The main reasoning methods identified in this section are safety analysis, Virtual Assessment of Automation in Field Operation (VAAFO), simulation and others. Safety analysis was used as a method for scenario analysis and simplification for reducing the number of test scenarios for HAV test and evaluation [104]. First, the concrete scenarios set are analyzed. Second, by analyzing concrete scenarios through the traversal of trajectories, trajectories that lead to collisions or test tasks uncompleted are obtained. By analyzing these trajectories, the SCPs (scenario characteristic parameters) of the corresponding scenario are obtained using functional decomposition [15], combined with fault tree analysis (FTA). By analyzing the overlap or relationship among the SCPs, the inclusion relationship among scenarios is obtained according to the SCPs included in different scenarios. By searching for the combination that contains the fewest scenarios but still covers all the SCPs, and using this set of scenarios to replace the original combination of test scenarios, the redundant evaluation scenarios were deleted. Using simulation as the method, the authors in [76] used public traffic data to calibrate SUMO (simulation tool, Simulation of Urban MObility) to perform traffic simulation. Based on the simulation results, the data is post-processed to extract concrete crash scenarios.

VAAFO [97] is used as a method in parallel with human driving for critical scenario identification. In VAAFO (Virtual Assessment of Automation in Field Operation), the vehicle is driven by a human. It receives information from all sensors, but is not connected to the actuators and instead operates in parallel while the human driver drives the vehicle. AD functions are running (simulating) within the perceived world. The trajectory planned by the AD function is compared with the human driving trajectory. The AD function may take a different decision than the human driver. This difference may consequently affect the behavior of other vehicles. The authors in [97] state that accident scenarios made by human drivers will be recorded by the police. Potential critical scenarios are filtered further to identify real critical scenarios after correction of the world model and criticality metrics. The VAAFO method will identify scenarios that are risky for the AD but not for human drivers.

As another method, the authors in [112] uses Long Short Term Memory (LSTM) networks to generate new collision data from prerecorded data. The data used to train the Recurrent Neural Networks (RNNs) comes from a simulation environment, but it could also make use of real accident data. In the example developed in the study, the data includes speed, the direction of vehicles and traffic light data. Once trained, the network can be used to generate new collision data starting from an initial seed that contains the initial speed, direction and traffic light state.

Criticality assessment: The criticality definitions of the articles in this section are also shown in TABLE 6. The criticality definition in [104] is based on safety analysis. This paper uses the term hazardous scenario, which refers to a scenario that may lead to harm, caused by the functional limitation or failures of the system. A scenario is critical if it is possible to have a collision in this scenario due to FuSa (Functional Safety) or a SOTIF problem with the ego vehicle. In [97], the criticality was defined via the assessment module, the simplest measure is whether a real collision has occurred or not, and the criticality definition also considered SOI. Therefore, it was also implementation-specific and related to a collision. The definitions in papers [76] and [112] do not consider a specific SoI. Yue et al. [76] defined a scenario risk index and Jenkins et al. [112] used RNN for criticality definition based on the collected data in simulation, and both are therefore non-implementation-specific.

6.3.3 Mechanisms for Coverage

Here we summarize and present our observations on how these articles evaluate the coverage results of the critical scenarios detected in their papers, all of which are based, of course, on the fact that the articles contain references to relevant content. Different methods also have different mechanisms for coverage definitions and evaluations. For all articles that use clustering methods, their coverage is based on the completeness of the feature vectors (predefined numerical features that represent some scenario variables or attributes), i.e., if these feature vectors are complete, then their clustered typical scenarios can well reflect and cover the scenario space represented by the defined scenario variables or attribute. The methods based on existing data (textual analysis and reconstruction&simulation) do not explicitly discuss the coverage results of the critical scenarios they found.

6.3.4 Required Information

All the required information in this section relates to sensor data, unstructured accident records, structured accident records, sentence template for NLP, simulator, FTA analysis, pre-crash scenario classes, traffic data and real driving sensor data. The involved databases are Korea National Police Agency Accident Database, NHTSA (National Highway Traffic Safety Administration), IGLAD [158] (Initiative for the Global Harmonization of Accident Data), Second Strategic Highway Research Program (SHRP2), China In-Depth Accident Study (CIDAS) and German In-Depth Accident Study (GIDAS).

6.4 Deductive Reasoning

This section focuses on the methods that use a deductive reasoning approach based on different sources of knowledge to find critical functional or logical scenarios. Based on this scope, a total of 6 related articles were identified.

Four of these papers [30], [61], [62], [63] focus on finding pre-crash scenarios by systematically considering all the possibilities under a set of explicitly pre-defined assumptions. The identified pre-crash scenarios can be used as safety-critical operational situations as introduced in Fig. 3. These four papers adopt a similar approach. As the first step, they take some available logical scenarios from function/system specification or safety analysis. Then, they define a structured road and initial conditions based on the function specification (e.g., 3 lane highway road) for the reasoning. Afterwards, the scenario is elaborated using the assumptions, e.g., by adding more vehicles and changing the vehicles' behaviors. The assumptions include: potential behaviors of vehicles on the road defined in the ODD, possible collision types, traffic rules, and function (SoI) features. All the identified pre-crash scenarios should be used for system verification and validation.

In addition to finding pre-crash scenarios as critical operational situations, there is also an approach to find complete critical scenarios, which also include a triggering condition. The authors in [64] proposed a method to identify and to quantify the risk of critical scenarios for highly automated driving vehicles, considering both functional insufficiencies and failures. They proposed a new method to combine triggering conditions with the fault tree to deduce critical scenario from a given safety goal. The approach can be summarized into the following steps: The first step is to simulate the automated driving functions as components and interfaces. The second step is to identify potential hazards related to the AD system. Using a HAZOP-like method, they first identify generic vehicle-level hazards that are independent of the underlying implementation. Second, they use a HAZOP-like approach again to identify Functional Insufficiencies with hazardous effects. Each AD function is treated as a black box. Next, an "environmental fault tree" is used to identify the causes of functional inadequacies and the corresponding environmental conditions for triggering a hazardous scenario.

Ponn et al. [114] proposed a methodology for an intelligent selection of relevant scenarios for the certification of automated vehicles. They proposed a two-stage optimization framework to generate concrete scenarios. A detailed optimization method was not proposed. In the first optimization stage, the parameters of Layer one, two and five (refer to the 6-layered model discussed in Sec. 2.3.1) are first optimized by sensor analysis and consideration of driving behavior. In addition, the trajectory of the potential conflict partner (Layer L4) is determined. In the second stage, further objects are defined (to refine the logical scenario) by considering the complexity, and their trajectories are optimized.

Criticality assessment: The criticality definitions of the articles in this section are all shown in TABLE 6. All the articles in this section relate to collision and are non-implementation specific. No implementation of a specific system is needed for these approaches. Basically, a critical scenario should contain at least one accident threat or collision (i.e., one accident type may potentially happen).

6.4.1 Mechanisms for Coverage

The coverage in relation to the methods of this section depends on a set of explicitly predefined assumptions in the scenario complexity definition (e.g., potential behaviors of vehicles on the road defined in ODD, possible collision types, traffic rules, and function (SoI) features).

6.4.2 Required Information

The required information in this section also relates to the assumptions, such as potential behaviors of vehicles on the highway, possible collision types, traffic rules, set of base pre-crash scenarios, set of predefined basic maneuvers, set of generic hazardous scenarios, function features, and system specification.

6.5 Finding Critical Scenes for CV-based Functions

This section focuses on the cluster of methods to find critical scenes to falsify a CV-based function. Based on the taxonomy in Fig. 10, methods in this cluster are summarized in Fig. 24 and elaborated in the following subsections.

6.5.1 Scene Representation

In this cluster, scenes can be represented in the following three ways:

Scene as an image: To align with the terms defined in Fig. 4, an executable scene is an image derived from a camera or stored in a database. Some methods in this cluster directly generate critical scenes as images [68], [115], [116], [124]. Some methods take images as inputs to explore for critical ones [125], [126].

Description-based: In this representation, a scene is described by a set of parameters [68], [125], [126], [127]. It can be used for both logical scenes and concrete scenes. A logical scene contains the ranges or the probability distributions of all the parameters. A concrete scene has a fixed value for each parameter. A tool is needed to transform a concrete scene into an image, based on, for instance, a video game (e.g., GTA V [68]) or a physics engine (e.g., Unreal engine [115]).

Image plus transformation: This is used to represent a transformed image. Examples of image transformation include shearing, blurring, adding occlusions, and changing weather conditions [119]. With this representation, a concrete scene is a transformed image, while a logical scenario is represented as an original image together with a parameterized transformation (e.g., shearing with different angles) [118], [119].

Different scene representations are required by different methods to find or to synthesize critical scenes. These methods are introduced in the next sub-section.

6.5.2 Critical Scene Generation or Exploration

As shown in Fig. 24, we identified the following three major types of methods to generate or explore a critical scene to falsify CV-based functions.

Image transformation: This type of methods assumes that a non-critical scene can be made more critical by changing or adding content on the image, for example, adding occlusions, intensifying the brightness, or changing weather into extreme conditions. Our literature search found the following image transformation approaches:

- **Direct image transformation:** In this method, the image transformation is applied directly to the image without the help of machine learning algorithms or optimization algorithm. In general, direct image transformation (e.g., translation, scaling, shearing, rotation) [118] is a simple technique. However, these transformations do not make the image more critical. They just generate different images, which in most of the cases are not realistic. Image transformation can also be performed by particular image processing tools. For example, Yang et al. [115] used Adobe Photoshop to add blur or rain effect through convolutional image transformation.
- **GAN method:** A Generative Adversarial Network (GAN) can be trained to generate new images that are similar but specifically different from the dataset [159]. In the context of CSI, GAN can help to automatically generate a large amount of driving scenes with various weather conditions, which look realistic [116], [124].
- **Optimization-based method:** Optimization algorithms can guide the exploration towards critical scenes, and also increase the coverage. For example, Pei et al. [118], [119] propose an optimization-based method to find a set of combinations of direct image transformations to increase the neuron coverage (i.e., the proportion of the neurons activated by a set of

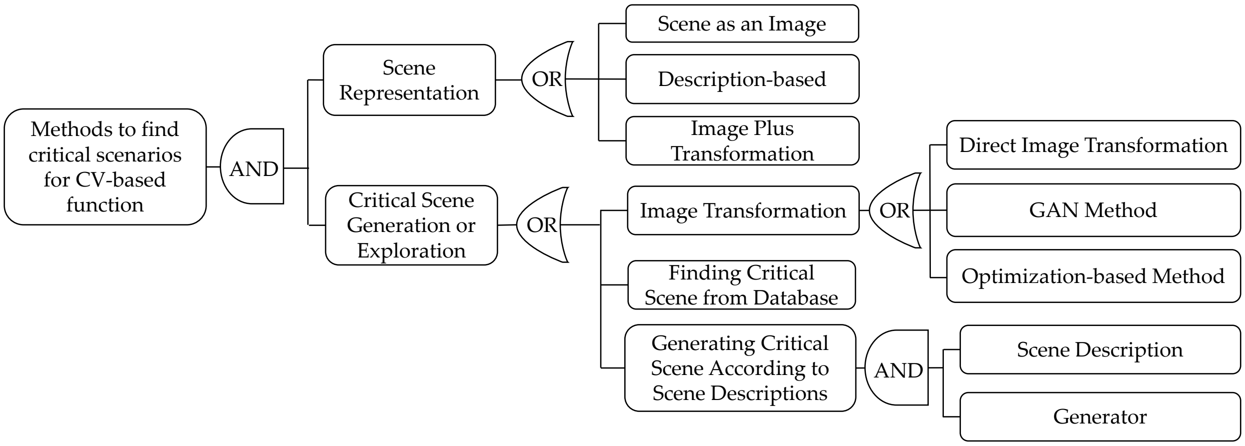
Methods to find critical scenarios for CV-based function
AND
Scene Representation
OR
Scene as an Image
Description-based
Image Plus Transformation
Critical Scene Generation or Exploration
OR
Image Transformation
OR
Direct Image Transformation
GAN Method
Optimization-based Method
Finding Critical Scene from Database
Generating Critical Scene According to Scene Descriptions
AND
Scene Description
Generator

Fig. 24. Methods to find critical scene for CV-based function

test inputs). Increasing neuron coverage improves the quality of a test suite [160].

Finding critical scene from database: This type of methods traverses all the images in a database to find the non-implementation-specific critical ones [125], [126], [127]. Images in the databases are firstly manually encoded into predefined parameterized scene descriptions. Thereafter, the core of this type of methods is the approach to assess the criticality of each scene according to its scene description. The criticality can be assessed based on the complexity of the scene using a pre-trained machine learning model. Discussion on criticality assessment methods in this cluster is given in Section 6.5.3.

Generating critical scene according to scene descriptions: This type of methods generates scenes according to a parameterized scene description with predefined semantics and syntax. The scene description may include environmental information such as weather conditions, light conditions and time. It may also include the information of road users such as the relative position, heading, dimension, car model, color, etc. According to these scene descriptions, images are generated with the help of a third-party tool, such as GTA V [68] and the Unreal Engine [115]. The generated scenes can support data augmentation (see examples in [68]), the verification of CV-based functions, and the identification of influential factors. Similar to the image transformation method, this method can also minimize the effort and budget compared to physically collecting these data. For instance, it can generate a potentially critical scene by adding specified occlusions, different weather conditions, or poor illumination.

6.5.3 Criticality Assessment

In this cluster, all the identified critical scenes are function-critical. They may cause unintended results of CV-based functions. Most papers associate critical scenes with challenging environment situations such as bad weather, occlusions, and various light conditions. The criticality can be assessed either directly on an image [68], [115], [116], [117], [118], [119], [127], [127], or on a scene description [125], [126]. Surrogate measures to assess the criticality of scenes are summarized in TABLE 6.

For the methods that directly assess criticality on images, most of them feed the image under assessment to the SoI (i.e., the function under verification) as input to assess its performance. Therefore, the assessed criticality is implementation-specific. In TABLE 6, surrogate measures for these methods include failures and differential behaviors, which are explained at the end of this sub-section. In this literature review, only one paper [127] was found to assess non-implementation-specific criticality on images. The corresponding surrogate measure for this method [127] in TABLE 6 is predictability, which is also explained at the end of this sub-section. It is categorized as non-implementation-specific since the criticality assessment process does not involve the SoI.

As discussed previously, a scene description is a list of parameters, each of which supports the description of one influential factor. Complexity in TABLE 6 is the surrogate measure to access the criticality of a scene based on all the parameter values in its description. Complexity intends to reflect how likely the performance of CV-based functions in general will be affected by this scene. Therefore, complexity is categorized as non-implementation-specific. Detailed explanation of complexity is elaborated at the end of this subsection.

The detail explanation of each surrogate measure in TABLE 6 for this cluster is as follows:

- **Failures:** It can help to find scenes that may lead the CV-based function to a specific failure mode (e.g., mis-detection or mis-classification) [68], [115], [116], [117]. This measure is used to find consequence-aware implementation-specific critical scenes.
- **Differential behaviors:** This category is used to assess the Deep Learning (DL) algorithms by measuring the number of differential behaviors (i.e., different outputs from multiple similar DL systems with the same input) [119] [118]. In safety-critical systems such as automated driving systems, differential behavior can lead to disastrous consequences (e.g., collision, crash) [119].
- **Complexity:** This category measures how challenging the scene is for the SoI. In the primary studies, a complex scene can be specified according to project

experience [68], [124]. The complexity of a scene can also be evaluated by a trained machine learning model. [125], [126]

- **Predictability:** For one frame in a video stream, its criticality can be assessed based on how difficult it can be predicted according to its previous frames. Bolte et al. [127] utilized an image prediction algorithm to predict each frame and compared the predicted frame with the real frame. The big difference implies low predictability. In their method, images with relevant objects (e.g., vehicles and pedestrians) and low predictability are considered critical.

6.5.4 Mechanisms for Coverage

Coverage is rarely discussed in the primary studies in this cluster. The only explicitly mentioned coverage is the neuron coverage (i.e., the proportion of neurons activated by the whole test set) [119], which assesses the test adequacy of neural networks. [160] This coverage is implementation-specific, and can not reflect the completeness of the identified scenes. A coverage defined based on the ODD can be a potential future research topic.

6.5.5 Required Information

We have identified general required information when generating or finding critical scene for CV-based function as follows:

Scene generator: The scene generator is used for description-based or image transformation-based method to generate a new scene. The scene generator can be semantic-descriptor tools, game editor (Unreal) [115], photo editor (Adobe) [119], etc.

Database: Most of the methods in this cluster required database. A critical scene can be synthesized according to an existing scene in a database [116], [118], [119], [124]. A database can also be used to train the machine learning model to access criticality of a new scene [125], [126].

Influential factors: The influential factors are required when specifying a critical scene description [68], [115], [117].

7 THE EVALUATION CATEGORY

This section mainly focuses on the evaluation category defined in Section 4 to answer the question ”How are the validity of the approach and the identified critical scenarios assessed?” for each cluster defined in Section 6. In the following, we structured this section according to the three subcategories of *Evaluation* in section 4 with each cluster.

7.1 Approach Verification

C1: Almost all the primary studies in this cluster perform case studies as the evaluation method to validate their proposed methods and the identified critical scenarios, where the studies in [48], [90] are in the absence of the validation part. Most of these case studies are realized by simulation. The study in [89] collects the real-world driving data for validation. In general, the validation in these papers shows that critical scenarios can be found using the proposed methods based on the defined criticality metrics.

C2: The evaluation methods in this category are realized by case studies and very similar to the ones of category C1.

C3: Inductive reasoning methods (Section 6.3) In this cluster 7 out of 12 articles described or assessed the availability and efficiency of the proposed approach. Since the papers in this category used either existing databases or available scenarios as data sources, the most used evaluation method is comparing the identified scenarios with the scenarios in the regulations. In the papers that generate functional or logical pre-crash scenarios by using clustering from real data, they directly compare these functional scenarios with the test scenario of the regulations or testing organizations, e.g., Euro NCAP ([54], [56], [113]). For other papers that generated concrete scenarios, the simulation results, like simulated damage or KPIs using the proposed methods, were compared directly with the recorded scenario or the test results.

C4: Deductive reasoning methods In this cluster 3 out of 6 articles evaluated the proposed approach by comparing the identified pre-crash scenario against the regulation or accident database. For example, the authors in [61] used various knowledge, such as possible collision types and traffic rules, to generate the pre-crash functional scenario and further evaluated the generated scenario catalog with respect to the naturalistic driving database and test regulations (Euro Ncap, UNECE, etc.).

C5: Finding critical scenes for CV-based functions Almost all the articles in this category used a case study as the evaluation method.

7.2 Criticality Validation

C1: In this category, there are no validations on the criticality of the scenarios themselves. In other words, these articles did not tackle whether the identified critical scenario in simulation is also critical in the real world, or the gap between real world and simulation.

C2: In this category, there are no validations on the criticality of the scenarios themselves.

C3: In this category, there are no validations on the criticality of the scenarios themselves.

C4: In this category, there are no validations on the criticality of the scenarios themselves.

C5: Similar to the papers in C1 and C2, there are no validations made to determine whether the identified critical scenario in simulation is also critical in the real world for the computer vision system. And, as discussed in Section C5, no such surrogate measure was discovered in the primary studies to identify critical scenes for the CV methods.

7.3 Coverage Assessment

C1: Regarding coverage evaluation, as shown in Fig. 20, different mechanisms, like N-wise or sampling size, were proposed in this cluster to assess the coverage, but there was no validation of the coverage results in real-world interpretation.

C2: Similar to the papers in C1, in this category, the coverage is defined by the exploration level of the given scenario space. The coverage is used to avoid local minimum [66] and promote the exploration in the scenario space [59].

C3: Inductive reasoning methods There are no direct definition and validation results of the coverage in this section. However, some articles validate their method by comparing the identified functional scenarios with the original database, which can be interpreted as a type of coverage assessment.

C4: Deductive reasoning methods The coverage in this section is defined by assumptions about the possible behavior of traffic participants (including ego vehicles) and the possible road topology. In these primary studies, these assumptions are mainly made involving layers L1 (urban and highway scenarios), L2 (traffic rule) and L4 of the 6-layer scenario description model (Table 2.. As aforementioned, coverage results of the deduced safety-critical operational situations were evaluated against the accident database or test regulations.

C5: Finding critical scenes for CV-based functions As mentioned in section 6.5.3, coverage is rarely discussed in the primary studies of this cluster. The only explicitly mentioned coverage is neuron coverage in [119]. This coverage is implementation-specific and cannot accurately reflect the completeness of the identified scenes.

8 Relation to Other Relevant Directions

In this section, we briefly present the other relevant directions discovered during the literature review but are out of scope of the survey. These relevant directions have important implications and relationships to the main idea of the CSI method. In the following section, we first give a brief introduction to these methods, then explain how they relate to our article, and finally list the differences.

8.1 Online risk assessment

Online risk assessment refers to the methods which evaluate and predict in real time if a situation could be dangerous and result in harm. As summarized in [161], these methods analyze the predicted trajectories of other vehicles and the potential colliding point. The methods presented in section 6.1.3 are mainly offline methods, whereas the methods here are for real-time vehicle application.

Relation: For both online and offline approaches, the identified influential factors as well as the critical scenario based on unexpected behaviors or events can be used. Second, the end-to-end scene risk prediction used in online risk assessment can also be used offline to determine a scene's criticality. Furthermore, risk indicators such as Time-To-Collision and Time-To-React can be used for both methods.

Differences: The online methods can only utilize the real street data whereas the CSI methods can utilize ground-truth environmental information. Offline methods, on the other hand, are not so easily transferred to online methods because the offline approach typically requires exhaustive searches, and computing all the potential trajectories of the vehicles is computationally costly hens not real-time capable.

8.2 Scenario-based Function Assessment

Function assessment is one of the main reasons for using scenarios in the development, verification and validation of ADSs. It is also one of the primary reasons for finding critical scenarios as discussed throughout this paper. Here we can consider both broad descriptions of scenarios, e.g., [36], [162], used for functional assessment, as well as different ways to achieve coverage of the scenario space in general, e.g., through Monte-Carlo simulations. Furthermore, Hauer et al. [90] suggest a method using a heuristic search and a fitness function to ensure that the scenarios specified are actually tested. It is also possible to do functional assessment using Monte-Carlo simulations of agent and simulation models [163]. In [163], these models are parameterized with different data sources to reflect actual driving scenarios.

Relation: As mentioned, function assessment is a common vein across many of the papers considered in this review, and one of the primary purposes for using scenarios in general is to achieve some kind of assessment of the function. This paper primarily focuses on the assessment of critical scenarios whereas the studies [36], [90], [162], [163] consider scenarios in a broader sense of AD function assessment. Having a "driver's license test" for ADSs would perhaps also be considered a function assessment. This aspect is however not considered in this review nor further in this section.

Differences: The primary difference to the approaches of this section is the lack of explicit discussion about how to find or identify critical scenarios. In [36], [162], scenarios are used as important descriptors to achieve function assessment, but there are no discussions about how to identify critical ones. Ensuring that specified scenarios are indeed tested, as described in [90], is surely relevant, but again outside of the scope of this review as it discloses no details with respect to finding critical scenarios. One could however note that the described methods could be used to ensure that critical as well as non-critical scenarios that are specified are executed. Monte-Carlo simulation, in general, is not covered in this review as Monte-Carlo only provides a brute force methodology for covering scenario space. However, importance sampling, as suggested in [164], is covered since it can be used to find the scenarios that are more critical to the system. The authors of [164] have also considered using subset simulation for accelerated testing of ADSs [165]. Similar to importance sampling, subset simulation guides the Monte-Carlo search for relevant scenarios using a KPI. If this KPI is chosen to capture a safety-critical aspect, this approach would also yield critical scenarios.

8.3 Scenario-based System Design

Scenarios have also been suggested for supporting system design. Authors in [166] outline a method to identify critical spots of scenario space (white spots) through simulation of an ADAS feature. These white spot areas are subsequently used to design the ADAS system such that it fulfills the requirements derived from the white spots identified in previous performance assessment(s). The paper [167] in a related vein, suggests the inverse, namely to define the ODD of the ADS using a world model of the intended use case for the ADS, where this world model is proposed to be constructed based on scenarios and populated using real traffic data. The paper [168] on the other hand suggests the opposite again, that is, to use simulation assessment as a means to determine the ODD of the ADS by removing road segments where the ADS perform poorly (c.f. white spots of [166]).

Relation: In all three cases above, scenarios are used (implicitly) to support the proposed process. Noteworthy is that both studies [166], [168] consider the parts of scenario space (in their respective definitions) that result in critical situations for the system being assessed.

Differences: Even though scenarios are considered, neither of the approaches discuss how to explicitly find the critical scenarios (white spots of [166]), which is the primary focus of this review. Thus, deploying the methods described in this review would increase the efficiency of [166], [168] in terms of needed resources for simulations.

8.4 Fault Injection

Fault Injection is a method that accelerates the occurrence of faults to test and evaluate faults in systems. It is an important part of the test process in the automotive standard ISO 26262 [16] and is distributed over the development stages. The author in [21] applied the fault injection method to find and mine situations and faults that can be dangerous to the vehicle's safety.

Relation: The target of article [21] is to find safety-critical situation and faults by using causal and counter-factual reasoning about the behavior of the ADS under a fault for safety analysis. The fault Injection method can therefore also be used to find critical scenario and situation.

Differences: The primary goal of the fault Injection is to evaluate the faults in system and analyze the system behavior based on the injected faults. The basic purpose of the fault Injection differs from that of the CSI method, but it is an interesting and well-established method for testing the system in autonomous driving applications.

8.5 Ontology Design and Influential Factor Analysis

The articles and methods here focus on finding and analyzing the influencing factor, and utilizing a domain ontology to capture and represent the environment of the system under test. As presented in Fig 3, a scenario condition can be modeled as a combination of several scenario factors. The main steps of the ontology design method in the section are similar to the method in the reasoning section regarding both deductive and inductive reasoning. They took input or knowledge from different sources, such as system specification, sensor error type, system knowledge or analysis like FTA, or FMEA analysis [169].

Relation: The articles in this section are not included in the main content. The reason is that they focus mainly on the use of ontology design to find triggering conditions or influencing factors, but these identified influencing factors can be further used in CSI method to find the critical scenarios.

Differences: The main difference between this section's approaches is that no further case study or application were performed to identify critical scenarios.

8.6 Formal Methods

Formal methods are used to define and verify unambiguous specifications of computer systems. These methods rely on a sufficiently complete abstract model of the real world and formal specifications that should resemble the informal requirements that should be imposed on the system. [170]

Relation: This literature review covers the papers which use formal methods to find counter examples as critical scenarios.

Differences: It is necessary to represent the relations between the behaviors of the agents and environmental conditions with hybrid automata, which need to be sophisticated enough to reflect reality and simple enough to guarantee computational feasibility.

8.7 Unknown Unknowns Detection in Computer Vision

Unknown unknowns in computer vision (CV) functions (e.g., object classification) refer to the case where the employed predictive model (e.g., an deep neural network) assign incorrect labels to instances with high confidence [171]. Explainable AI [172] approaches can be adopted to detect and avoid unknown unknowns.

Relation: These unknown unknowns are typically caused by the mismatch between training data and the cases encountered at test time. Since the CV functions are almost blind to such errors, they can be considered as implementation-specific critical scenes.

Differences: According to the authors' understanding, these approaches should be considered in Section 6.5. However, no automated driving applications of these approaches were found during our literature search.

8.8 Data Augmentation

Data augmentation is used for machine learning based functions to expand the size of the training set by generating new training data.

Relation: Section 6.5 covers methods to generate new scenes according to a predefined scene description. These generated scenes can also be used for data augmentation.

Differences: Traditional data augmentation methods focus on how to generate a big amount of different scenes as training data. Methods introduced in Section 6.5 focus on generating new and also potentially critical scenes.

9 Discussion

This section discusses how the results of the literature review (as presented in Sections 4, 5, 6 and 7) answer the three research questions defined in Section 3.1, repeated below.

RQ1: What would be a taxonomy that allows to systematically categorize and compare state-of-the art CSI methods for ADS and ADAS?

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

RQ2: What is the current status of CSI methods research with respect to this taxonomy?

RQ3: What are the remaining problems and challenges for further investigation?

We have addressed RQ1 by analyzing relevant industrial standards, as discussed in Section 2, and the primary studies collected with the methodology introduced in Section 3. The taxonomy was derived through iterations and validated by applying it to the collected studies. The proposed taxonomy, as presented in Section 4, can sufficiently categorize all the primary studies. To support further research and engineering in areas related to critical scenarios and their identification methods, this taxonomy can help researchers to provide a contextual understanding - the big picture of such methods-, and provide guidance regarding CSI method design and adoption.

RQ2 is answered in Sections 5, 6 and 7 by analyzing all the primary studies according to the taxonomy. The answer to this question is also used as the basis to answer RQ3.

RQ3 is answered by internal discussions among all the authors; ”remaining problems and challenges” are presented in Sections 5, 6 and 7. Some important points are summarized and discussed in the rest of this section, including suggestions for future work.

9.1 Coverage

Within the context of this paper, coverage can be defined in 3 ways (referred to as coverage types in the following): 1) the coverage of the exploration with respect to the given scenario space; 2) the coverage of all the critical scenarios in the given scenario space (i.e., the proportion of the identified critical scenarios among all the critical scenarios within the given scenario space); and 3) the coverage of all the critical functional insufficiencies and their triggering conditions under a given functional scenario or an ODD. The type 1 coverage evaluates the exploration of the scenario space. The type 2 coverage evaluates the effectiveness of a *logical* $\rightarrow$ *concrete* CSI method. As shown in Fig. 6, identified critical scenarios are used to support the identification of critical functional insufficiencies so as to improve the safety of the intended functionalities. Therefore, the type 3 coverage is essential for safety analysis. These three types of coverage are discussed in the rest of this section.

The type 1 coverage is valid for the CSI methods finding critical concrete scenarios from a given logical scenario. Most of these methods are covered in clusters C1, C2 and C5. Due to the huge size of the scenario space, reaching full coverage is practically impossible. To quantify the level of coverage, coverage metrics can be defined. Adopted coverage metrics include sampling size (for sample-based exploration methods) and combinatorial coverage (for combinatorial testing methods). It is difficult to directly measure the type 2 and type 3 coverage. The type 2 coverage can be indirectly reflected by the coverage metrics defined for the type 1 coverage. As discussed before, the type 3 coverage is of the most importance. However, the relationships between these three types of coverage have not been sufficiently discussed in the primary studies. Understanding these relationships can help to determine a sufficient coverage level (in terms of a particular coverage metric) on the scenario space. It can also help to analyze the completeness of the safety analysis. These relationships can be discussed from the perspective of how critical scenarios may be missed.

Relation between type 1 and type 2 coverage: With potential limitations of an adopted exploration method, it may not be possible to identify all the critical concrete scenarios in a given scenario space (i.e., the given logical scenario). According to Fig. 16, these limitations can stem from the coverage level employed by the instantiation process or the criticality assessment method. Since full coverage is not reached, some critical scenarios may not be covered by the specified coverage level. Even though a critical scenario is reached during the exploration, it may still be assessed as noncritical due to the limitation of the criticality assessment method. If a simulator is used for criticality assessment, its fidelity (i.e. the influential factors in the simulator can represent those factors in real world) will affect the accuracy of the criticality assessment. Increasing the fidelity of simulators is one way to solve this problem. However, a high-fidelity simulator normally entails high computational resources, and thereby increases power and time consumption. Another way to solve this problem is to design surrogate criticality measures that can tolerate some simulation error (i.e. the differences between simulation and real results). For example, instead of collision, Time To Collision (TTC) can be used to find more potentially critical scenarios. However, as analyzed in Section 6.5.3 and shown in Table 6, no such surrogate measure was found in the primary studies to find critical scenes for the CV methods. On the other hand, some critical scenarios may also be filtered out by the limitation of the adopted surrogate criticality measure. For example, if longitudinal TTC is used as the surrogate measure, critical scenarios caused by lateral collision will be missed. Therefore, an appropriate approach to determine the criticality measure can be another future research direction. In addition, another way to increase the fidelity of the simulator is to bring the vehicle into the loop of the simulation to combine the real world and the virtual world, such as the methods introduced by Junietz et al. [97], Feng et al. [85] and Li et al. [173].

Relation between type 2 and type 3 coverage: Even if all the critical concrete scenarios within the given logical scenario are identified, one can still not claim a full coverage of the type 3 coverage under the corresponding functional scenario. Due to the potential misalignment between the logical scenario and the functional scenario, there might be critical scenarios that are not covered by the logical scenario. The reason for the misalignment is the assumptions made when formalizing the functional scenario (referring to Fig. 4). For example, some influential factors might be missing, or the models of some influential factors might be too simplified (e.g., using a constant speed model to represent the behavior of another vehicle). blueThis misalignment is part of the specification gap defined by Stellet et al. [174]. Section 8.5 briefly discusses how to find and formulate influential factors. Even though it is not the focus of this survey paper, it is an important topic for the safety analysis in terms of SOTIF. A literature review on influential factor identification could be a valuable future work.

Instead of being identified from critical scenarios, functional insufficiencies can also be derived by analyzing the

AD functions with safety analysis methods, such as the ones introduced in [175] and [64]. As inspired by traditional safety engineering [176], the coverage of the derived functional insufficiencies can be increased by conducting both top-down (e.g., fault tree analysis) and bottom-up (e.g., Failure Modes and Effects Analysis (FMEA)) safety analysis methods, where the top refers to the vehicle level safety goals, and the bottom refers to the malfunctioning behaviors of the components together with the corresponding triggering conditions. To this end, a potential valuable future research direction could be to propose a methodology that combines these safety analysis approaches and the CSI approaces introduced in this paper.

In addition, as discussed in Section 6.1.4, some primary studies assume that close (in terms of the distance on the scenario space) critical scenarios are likely to reflect the same functional insufficiency with the same triggering condition. To this end, increasing the diversity of the identified critical scenarios can increase the coverage of the identified functional insufficiencies. However, explicit coverage metrics for the identified functional insufficiencies and triggering conditions have not been found in the primary studies.

When systematically deducing safety-critical operational situations (methods in cluster C4), the coverage is defined by the assumptions made about the possible behaviors of the involved traffic participants (including the ego vehicle) and the possible road topology. In the primary studies, these assumptions are made on layers L1 and L4 of the 6-layer scenario description model in Table 2, since they all consider highway scenarios. For urban scenarios, the layer L2 in the 6-layer model (e.g., traffic lights) should also be considered. In addition, as discussed in Section 7, coverage of the deduced safety-critical operational situations can be evaluated against an accident database.

Corresponding to Fig 3, to achieve a complete safety analysis, coverage needs to be explicitly defined for hazards, malfunctioning behaviors, triggering conditions and safety-critical operational situations. As a precondition, this entails an explicit coverage definition for all the influential factors. These factors are classified into layers as explained in Table 2. A more detailed classification of these factors in each layer is necessary to facilitate the systematic analysis of both safety-critical operational situations and triggering conditions. Therefore, this would represent another relevant future research topic.

9.2 The ALARP Principle

Identifying all the functional insufficiencies with all the triggering conditions is practically impossible. The iteration in Fig. 6 should be governed by the ALARP principle (i.e. to guarantee that the risk of harm is ”As Low As Reasonably Practicable”), see e.g., [176]. The core concept behind ”reasonably practicable” is to control the risk of harm to an expected level. Therefore it is necessary to know how to control risk; and how to determine the expected level.

ISO 26262 [16] answers these two questions for the risk caused by systematic faults and random hardware faults. In ISO 26262, the risk of harm is determined by the severity and the probability of the harm. The ALARP principle in ISO 26262 is to control the residual risk (the risk remaining after the deployment of all the safety measures) to a reasonable level.

For the harm caused by functional insufficiency, risk can be defined based on the severity of the harm and the probability of exposure of the safety-critical scenario. In some of the primary studies, this probability of exposure is estimated according to the historical traffic data of human-driven vehicles. However, automated driving vehicles may exhibit different driving behaviors. Therefore, human-driving data may not precisely reflect the situations with automated driving vehicles. To this end, an effective approach to estimate the risk of harm related to functional insufficiencies represents a future research direction. In particular, given the novelty and unknowns of automated driving, this emphasizes the important role of continuous data gathering to support methodology improvements. An ability to better estimate the risk of harm would can support the determination of the stop condition of the iterative process illustrated in Fig. 6.

9.3 Scenario Space Explosion

Due to the openness of the driving environment, a vast amount of influential factors compose an ODD to explore. The exploration effort can be reduced by ”divide and conquer” approaches. To our understanding, the scenario space can be partitioned from two perspectives.

As suggested in [167], the first perspective is to divide an ODD into applicable use cases. Each use case is a subspace of the ODD represented as one functional scenario. In this way, critical scenarios are identified for each use case, instead of the whole ODD. The challenge of this method is to guarantee that the union set of all the use cases is equivalent to the whole ODD.

Different SoIs provide a second perspective for partitioning a scenario space (i.e. a logical ODD). An SoI can be either the whole ADS or a particular automated driving function.[12] Triggering conditions for different automated driving functions contain different influential factors [15]. For example, the angle of the sun may affect some perception functions, but it will not affect the vehicle dynamic control functions. Therefore, the number of influential factors for a particular AD function is much lower than that for the whole ADS. To this end, the exploration effort of a CSI method can be reduced by finding triggering conditions for each individual AD function. The identified triggering conditions for all the AD functions can be systematically combined to support the final safety analysis on the vehicle level. However, the propagation of a functional insufficiency is not deterministic due to the resilience of the downstream AD functions [21]. For example, if a vehicle is not detected in some individual camera frames by the computer vision function, this can be resolved by the following object tracing and sensor fusion algorithms for most cases. Therefore, to have a complete safety analysis, it is necessary to explicitly analyze the resilience of the system (i.e. to identify the scenarios where unintended behaviors from upstream functions cannot be resolved), especially to analyze the resilience of the object

[12] The relations between ADS level testing and function level testing are also discussed as ”scenario-based testing v.s. functionality-based testing”. [173], [177]

tracking functions and the sensor fusion functions. According to this literature review, only one paper [57] was found to indirectly touch this topic. Investigating resilience as just described, would thus provide a useful research direction. Moreover, a systemic view would in addition be required to ensure that the partitioning into the SoIs - and the ensuing composition of ”evidence” is complete, for example with respect to common cause failures. As discussed in this section this requires new methods that combine different approaches to critical scenarios identification, coverage and validation.

9.4 Other Sources of Harm

Fig. 3 shows the sources of harm that are considered in this paper. Besides these, there are also other aspects that can threaten the safety of an ADS, such as the ones listed below. For a complete safety analysis, all the sources of harm should be considered together, since a combination of multiple sources of harm may lead to a new hazardous event.

Potential lack of Communication: As shown in [178], some functional insufficiencies cannot be resolved without V2V (Vehicle to Vehicle) and V2I (Vehicle to Infrastructure) communication. Examples of such functional insufficiencies include occlusions, traffic violation by other road users, and the uncertainty of behavior prediction.

Cyberattack: An ADS can be hacked, especially when V2X (vehicle to everything) communication is adopted. Standard ISO/SAE 21434 [14] discusses how to mitigate the risk regarding cybersecurity.

Misuse: Misuse is considered in ISO/PAS 21448 [10] but not emphsized in this paper. It can be considered as part of a scenario on layer L0 in Table 2.

Problems outside the embedded system: ISO 26262 [16] and ISO/PAS 21448 [10] only focus the the safety issues caused by the embedded system. Other vehicle failures, such as a flat tire, a broken suspension or a battery fire, can also lead to significant harms.

10 Conclusion

For assuring safety of autonomous and automated driving, it is essential to be able to efficiently and effectively derive critical scenarios, i.e. situations that cause potential risks of harm (safety risks). Such critical scenarios need explicit consideration in ADS design and V&V efforts. Moreover, the use of state of the art techniques for CSI promises to reduce the V&V space by focusing on critical scenarios.

In this paper, we contribute to the challenge of safety and quality assurance of ADS and ADAS. In particular, we presented the results obtained from a systematic mapping study in the area of CSI methods. Moreover, we introduced a taxonomy focusing on practitioners for supporting them in selecting the right CSI method for identifying critical scenarios according to their project-specific needs. The introduced survey covers assumptions, levels of abstraction, metrics stating criticality, methods for the generation of critical scenarios and coverage measures.

In addition, we discussed research gaps obtained after analyzing the primary studies, for example the coverage metrics, the criticality measures, the identification and classification of influential factors, and the lack of a methodological framework to combine different CSI methods and also to connect the CSI methods with other safety analysis processes. Hence this survey may also be considered helpful in providing guidance regarding future research directions.

Acknowledgments

The authors of this paper have received support from a number of initiatives including: TECoSA - the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH Royal Institute of Technology and the ECSEL projects InSecTT (grant agreement No 876038 - www.insectt.eu) and ArchitectECA2030 (grant agreement No 877539 - https://autoc3rt.automotive.oth-aw.de/). The ECSEL projects have received funding from the ECSEL Joint Undertaking which receives support from the European Union’s Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. Finally, support is also acknowledged from the CD-Laboratory for Quality Assurance Methodologies for Autonomous Cyber-Physical Systems (QAMCAS) (https://qamcas.ist.tugraz.at/) with financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association.

Disclaimer

The document reflects only the view of the authors. The European Commission and the companies (i.e. Sigma Technology Consulting, Scania, AVL List GmbH, TU Graz and Zenseact) are not responsible for any use that may be made of the information it contains.

References

[1] M. Törngren, “Cyber-physical systems have far-reaching implications,” *HiPEAC Vision 2021*, Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4710500

[2] E. Frazzoli, “Can we put a price on autonomous driving?” *MIT Technology Review*, March 2014.

[3] C. F. Kerry and J. Karsten, “Gauging investments in self-driving cars,” 2017.

[4] A. Shetty, M. Yu, A. Kurzhanskiy, O. Grembek, H. Tavafoghi, and P. Varaiya, “Safety challenges for autonomous vehicles in the absence of connectivity,” *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103133, 2021.

[5] N. Kalra and S. M. Paddock, “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?” *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[6] On-Road Automated Driving (ORAD) committee, “J3016 Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” SURFACE VEHICLE RECOMMENDED PRACTICE, Tech. Rep., 2021.

[7] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a Formal Model of Safe and Scalable Self-driving Cars,” *arXiv:1708.06374 [cs, stat]*, 2017, arXiv: 1708.06374.

[8] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, “Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, vol. 2015-Octob. IEEE, sep 2015, pp. 982–988.

[9] ASAM, "ASAM: STANDARDIZATION FOR HIGHLY AUTOMATED DRIVING SIM:Guide," Association for Standardization of Automation and Measuring Systems, Tech. Rep., 2021. [Online]. Available: https://www.asam.net/asam-guide-simulation/

[10] International Organization for Standardization, "Road vehicles - Safety of the intended functionality," 2019.

[11] Underwriters Laboratories, Northbrook, USA, "UL 4600:2020 – Standard for Evaluation of Autonomous Products," 2020.

[12] IEEE Committee VT/ITS - Intelligent Transportation Systems, "P2846 - Assumptions for Models in Safety-Related Automated Vehicle Behavior - Under development," 2021.

[13] "Iso/awi ts 5083: Road vehicles — safety for automated driving systems — design, verification and validation - under development," 2021.

[14] "Iso/sae 21434 - road vehicles — cybersecurity engineering (under development)," 2021.

[15] C. Amersbach and H. Winner, "Functional decomposition—A contribution to overcome the parameter space explosion during validation of highly automated driving," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S52–S57, jun 2019.

[16] International Organization for Standardization, "ISO 26262: Road vehicles–Functional safety," 2018.

[17] The British Standards Institution, "Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification," 2020.

[18] International Organization for Standardization, "Road vehicles — Terms and definitions of test scenarios for automated driving systems," 2021.

[19] ——, "Road vehicles — Engineering framework and process of scenario-based safety evaluation," 2021.

[20] ——, "Road vehicles — Taxonomy for operational design domain for automated driving systems," 2021.

[21] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 112–124.

[22] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[23] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

[24] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, nov 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7368032/

[25] J. Xu, Q. Luo, K. Xu, X. Xiao, S. Yu, J. Hu, J. Miao, and J. Wang, "An automated learning-based procedure for large-scale vehicle dynamics modeling on baidu apollo platform," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5049–5056.

[26] B. Giesler, "An Appeal Practical Standards in Autonomous Driving," *ATZelektronik worldwide*, vol. 12, no. 2, pp. 16–21, apr 2017.

[27] M. Törngren, X. Zhang, N. Mohan, M. Becker, L. Svensson, X. Tao, D.-J. Chen, and J. Westman, "Architecting safety supervisors for high levels of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1721–1728.

[28] J. M. Carroll, *Scenario-based design: envisioning work and technology in system development*. John Wiley & Sons, Inc., 1995.

[29] H. Winner, K. Lemmer, T. Form, and J. Mazzega, "Pegasus—first steps for the safe introduction of automated driving," in *Road Vehicle Automation 5*. Springer, 2019, pp. 185–195.

[30] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki, and L. Eckstein, "A framework for definition of logical scenarios for safety assurance of automated driving," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S65–S70, jun 2019.

[31] P. Koopman, A. Kane, and J. Black, "Credible autonomy safety argumentation," in *27th Safety-Critical Sys. Symp. Safety-Critical Systems Club, Bristol, UK*, 2019.

[32] C. Neurohr, L. Westhofen, T. Henning, T. de Graaff, E. Möhlmann, and E. Böde, "Fundamental considerations around scenario-based testing for automated driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 121–127.

[33] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE access*, vol. 8, pp. 87 456–87 477, 2020.

[34] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Citeseer, Tech. Rep., 2007.

[35] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571–583, 2007.

[36] J. Bach, S. Otten, and E. Sax, "Model based scenario specification for development and test of automated driving functions," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-Augus, no. Iv, pp. 1149–1155, 2016.

[37] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.

[38] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks?" *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 746–753, 2017.

[39] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, may 2013.

[40] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. Gaspar Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica, and H. Felbinger, "Finding critical scenarios for automated driving systems: The data extraction form," KTH, Tech. Rep., 2021. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-302116

[41] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and software technology*, vol. 117, p. 106200, 2020.

[42] E. De Gelder and J. P. Paardekooper, "Assessment of Automated Driving Systems using real-life scenarios," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 589–594, 2017.

[43] F. Reiterer, J. Zhou, J. Kovanda, V. Rulc, V. Kemka, and L. del Re, "Beyond-Design-Basis Evaluation of Advanced Driver Assistance Systems," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, jun 2019, pp. 2119–2124.

[44] Z. Huang, H. Lam, and D. Zhao, "Sequential experimentation to efficiently test automated vehicles," in *2017 Winter Simulation Conference (WSC)*. IEEE, dec 2017, pp. 3078–3089.

[45] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 3, pp. 595–607, 2016.

[46] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally Efficient Safety Falsification of Adaptive Cruise Control Systems," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 2879–2886, 2019.

[47] N. Li, I. Kolmanovsky, and A. Girard, "Model-free optimal control based automotive control system falsification," in *2017 American Control Conference (ACC)*, may 2017, pp. 636–641.

[48] S. Khastgir, G. Dhadyalla, S. Birrell, S. Redmond, R. Addinall, and P. Jennings, "Test Scenario Generation for Driving Simulators Using Constrained Randomization Technique," in *SAE Technical Paper*. sae.org, mar 2017.

[49] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*, 2018.

[50] F. Kluck, M. Zimmermann, F. Wotawa, and M. Nica, "Genetic Algorithm-Based Test Parameter Optimization for ADAS System Testing," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, jul 2019, Conference Paper, pp. 418–425.

[51] F. Klück, M. Zimmermann, F. Wotawa, and M. Nica, "Performance comparison of two search-based testing strategies for adas system validation," in *IFIP International Conference on Testing Software and Systems*. Springer, 2019, pp. 140–156.

[52] H. Beglerovic, M. Stolz, and M. Horn, "Testing of autonomous vehicles using surrogate models and stochastic optimization," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-March, pp. 1–6, 2018.

[53] X. Wang, H. Peng, and D. Zhao, "Combining reachability analysis and importance sampling for accelerated evaluation of highway automated vehicles at pedestrian crossing," *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 1, 2021.

[54] W. Hu, X. Xu, Z. Zhou, Y. Liu, Y. Wang, L. Xiao, and X. Qian, "Mining and comparative analysis of typical pre-crash scenarios from iglad," *Accident Analysis & Prevention*, vol. 145, p. 105699, 2020.

[55] Y. Kim, S. Tak, J. Kim, and H. Yeo, "Identifying major accident scenarios in intersection and evaluation of collision warning system," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[56] B. Sui, N. Lubbe, and J. Bärgman, "A clustering approach to developing car-to-two-wheeler test scenarios for the assessment of automated emergency braking in china using in-depth chinese crash data," *Accident Analysis & Prevention*, vol. 132, p. 105242, 2019.

[57] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, 2019.

[58] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive Stress Testing for Autonomous Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2018-June. IEEE, jun 2018, pp. 1–7.

[59] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 163–168.

[60] M. Koren and M. J. Kochenderfer, "Efficient Autonomy Validation in Simulation with Adaptive Stress Testing," in *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*. IEEE, oct 2019, pp. 4178–4183.

[61] J. Zhou and L. del Re, "Reduced complexity safety testing for adas & adf," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5985–5990, 2017.

[62] L. Huang, Q. Xia, F. Xie, H.-L. Xiu, and H. Shu, "Study on the Test Scenarios of Level 2 Automated Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, jun 2018, pp. 49–54.

[63] F. Xie, T. Chen, Q. Xia, L. Huang, and H. Shu, "Study on the Controlled Field Test Scenarios of Automated Vehicles," in *SAE Technical Papers*, vol. 2018-Augus, no. August, aug 2018.

[64] B. Kramer, C. Neurohr, M. Büker, E. Böde, M. Fränzle, and W. Damm, "Identification and quantification of hazardous scenarios for automated driving," in *International Symposium on Model-Based Safety and Assessment*. Springer, 2020, pp. 163–178.

[65] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1555–1562.

[66] C. E. Tuncali and G. Fainekos, "Rapidly-exploring Random Trees for Testing Automated Vehicles," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 661–666, 2019.

[67] F. Batsch, A. Daneshkhah, M. Cheah, S. Kanarachos, and A. Baxendale, "Performance Boundary Identification for the Evaluation of Automated Vehicles using Gaussian Process Classification," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, oct 2019, pp. 419–424.

[68] D. Fremont, X. Yue, T. Dreossi, S. Ghosh, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: Language-based scene generation," *arXiv preprint arXiv:1809.09310*, 2018.

[69] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2019-June, no. Iv, pp. 2352–2358, 2019.

[70] M. Althoff and S. Lutz, "Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, no. Iv, pp. 1326–1333, 2018.

[71] C. E. Tuncali, S. Yaghoubi, T. P. Pavlic, and G. Fainekos, "Functional gradient descent optimization for automatic test case generation for vehicle controllers," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, vol. 2017-Augus. IEEE, aug 2017, pp. 1059–1064.

[72] L. Li, Y.-L. Lin, N.-N. Zheng, F.-Y. Wang, Y. Liu, D. Cao, K. Wang, and W.-L. Huang, "Artificial intelligence test: a case study of intelligent vehicles," *Artificial Intelligence Review*, vol. 50, no. 3, pp. 441–465, oct 2018.

[73] L. Li, N. Zheng, and F.-Y. Wang, "A Theoretical Foundation of Intelligence Testing and Its Application for Intelligent Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6297–6306, oct 2021.

[74] P. Koopman and F. Fratrik, "How Many Operational Design Domains, Objects, and Events?" in *Proceedings of AAAI Workshop on Artificial Intelligence Safety*, Honolulu, USA, jan 2019.

[75] L. Stark, M. Düring, S. Schoenawa, J. E. Maschke, and C. M. Do, "Quantifying Vision Zero: Crash avoidance in rural and motorway accident scenarios by combination of ACC, AEB, and LKS projected to German accident occurrence," *Traffic Injury Prevention*, vol. 20, no. sup1, pp. S126–S132, 2019.

[76] B. Yue, S. Shi, S. Wang, and N. Lin, "Low-Cost Urban Test Scenario Generation Using Microscopic Traffic Simulation," *IEEE Access*, vol. 8, pp. 123 398–123 407, 2020.

[77] C. Neurohr, L. Westhofen, M. Butz, M. H. Bollmann, U. Eberle, and R. Galbas, "Criticality Analysis for the Verification and Validation of Automated Vehicles," *IEEE Access*, vol. 9, no. i, pp. 18 016–18 041, 2021.

[78] J. Tao, Y. Li, F. Wotawa, H. Felbinger, and M. Nica, "On the industrial application of combinatorial testing for autonomous driving functions," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2019, pp. 234–240.

[79] F. Klück, Y. Li, M. Nica, J. Tao, and F. Wotawa, "Using ontologies for test suites generation for automated and autonomous driving functions," in *2018 IEEE International symposium on software reliability engineering workshops (ISSREW)*. IEEE, 2018, pp. 118–123.

[80] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, mar 2018.

[81] M. Nabhan, M. Schoenauer, Y. Tourbier, and H. Hage, "Optimizing coverage of simulated driving scenarios for the autonomous vehicle," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, nov 2019, pp. 1–5.

[82] Y. Akagi, R. Kato, S. Kitajima, J. Antona-Makoshi, and N. Uchida, "A Risk-index based Sampling Method to Generate Scenarios for the Evaluation of Automated Driving Vehicle Safety *," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, oct 2019, pp. 667–672.

[83] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part i: Methodology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1573–1582, 2020.

[84] S. Feng, Y. Feng, H. Sun, S. Bao, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part ii: Case studies," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[85] S. Feng, Y. Feng, X. Yan, S. Shen, S. Xu, and H. X. Liu, "Safety assessment of highly automated driving systems in test tracks: a new framework," *Accident Analysis & Prevention*, vol. 144, p. 105664, 2020.

[86] S. Feng, Y. Feng, H. Sun, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles: An adaptive framework," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[87] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[88] W. Huang, Y. Lv, L. Chen, and F. Zhu, "Accelerate the autonomous vehicles reliability testing in parallel paradigm," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 922–927.

[89] D. Stumper and K. Dietmayer, "Towards Criticality Characterization of Situational Space," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, 2018, pp. 3378–3382.

[90] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2019, pp. 69–84.

[91] S. Wagner, A. Knoll, K. Groh, T. Kühbeck, D. Watzenig, and L. Eckstein, "Virtual assessment of automated driving: Methodology, challenges, and lessons learned," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 12-02-04-0020, pp. 263–277, 2019.

[92] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. von Vietinghoff, and T. Pfitzer, "Experience Paper: Search-Based Testing in Automated Driving Control Applications," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, nov 2019, pp. 26–37.

[93] P. Junietz, F. Bonakdar, B. Klamann, and H. Winner, "Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, 2018.

[94] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe, and K. Eder, "An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification," in *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, aug 2020, pp. 31–38.

[95] S. Wagner, K. Groh, T. Kuhbeck, M. Dorfel, and A. Knoll, "Using Time-to-React based on Naturalistic Traffic Object Behavior for Scenario-Based Risk Assessment of Automated Driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2018-June, no. Iv. IEEE, jun 2018, pp. 1521–1528.

[96] P. Du and K. Driggs-Campbell, "Finding Diverse Failure Scenarios in Autonomous Systems Using Adaptive Stress Testing," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 4, pp. 12–02–04–0018, dec 2019.

[97] P. Junietz, W. Wachenfeld, V. Schönemann, K. Domhardt, W. Tribelhorn, and H. Winner, "Gaining Knowledge on Automated Driving's Safety—The Risk-Free VAAFO Tool," in *Lecture Notes in Control and Information Sciences*. Springer International Publishing, 2019, vol. 476, pp. 47–65.

[98] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 1961–1967, 2019.

[99] S. Kuutti, S. Fallah, and R. Bowden, "Training adversarial agents to exploit weaknesses in deep control policies," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 108–114.

[100] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 8271–8277, 2019.

[101] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 9849–9860.

[102] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2019*. ACM Press, 2019, pp. 257–267.

[103] ——, "Automatically reconstructing car crashes from police reports for testing self-driving cars," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 290–291.

[104] Y. Qi, K. Li, W. Kong, Y. Wang, and Y. Luo, "A trajectory-based method for scenario analysis and test effort reduction for highly automated vehicle," *SAE Technical Papers*, vol. 2019-April, no. April, pp. 1–8, 2019.

[105] X. Qin, N. Aréchiga, A. Best, and J. Deshmukh, "Automatic testing and falsification with dynamically constrained reinforcement learning," *arXiv preprint arXiv:1910.13645*, 2019.

[106] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, "Using control synthesis to generate corner cases: A case study on autonomous driving," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, 2018.

[107] J. Zhou and L. del Re, "Safety Verification Of ADAS By Collision-free Boundary Searching Of A Parameterized Catalog," in *2018 Annual American Control Conference (ACC)*, jun 2018, pp. 4790–4795.

[108] S. Cutrone, C. W. Liew, B. Utter, and A. Brown, "A Framework for Identifying and Simulating Worst-Case Animal-Vehicle Interactions," in *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, 2019, pp. 1995–2000.

[109] X. Ma, Z. Ma, X. Zhu, J. Cao, and F. Yu, "Driver Behavior Classification under Cut-In Scenarios Using Support Vector Machine Based on Naturalistic Driving Data," in *WCX SAE World Congress Experience*. SAE International, apr 2019.

[110] V. Bithar and A. Karumanchi, "Application of collision probability estimation to calibration of advanced driver assistance systems," *SAE Technical Papers*, vol. 2019-April, no. April, 2019.

[111] S. Masuda, H. Nakamura, and K. Kajitani, "Rule-based searching for collision test cases of autonomous vehicles simulation," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1088–1095, 2018.

[112] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, and J. Schroeder, "Accident Scenario Generation with Recurrent Neural Networks," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, 2018, pp. 3340–3345.

[113] J. J. So, I. Park, J. Wee, S. Park, and I. Yun, "Generating Traffic Safety Test Scenarios for Automated Vehicles using a Big Data Technique," *KSCE Journal of Civil Engineering*, vol. 23, no. 6, pp. 2702–2712, jun 2019.

[114] T. Ponn, C. Gnandt, and F. Diermeyer, "An optimization-based method to identify relevant scenarios for type approval of automated vehicles," in *Proceedings of the ESV—International Technical Conference on the Enhanced Safety of Vehicles, Eindhoven, The Netherlands*, 2019, pp. 10–13.

[115] S. Yang, W. Deng, Z. Liu, and Y. Wang, "Analysis of illumination condition effect on vehicle detection in photo-realistic virtual world," SAE Technical Paper, Tech. Rep., 2017.

[116] H. Yu and Xin Li, "Intelligent corner synthesis via cycle-consistent generative adversarial networks for efficient validation of autonomous driving systems," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 9–15.

[117] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems wit machine learning components," *J. Autom. Reason.*, vol. 63, no. 4, pp. 1031–1053, 2019.

[118] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, may 2018, pp. 303–314.

[119] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[120] F. Gao, J. Duan, Y. He, and Z. Wang, "A Test Scenario Automatic Generation Strategy for Intelligent Driving Systems," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–10, jan 2019.

[121] Q. Xia, J. Duan, F. Gao, T. Chen, and C. Yang, "Automatic Generation Method of Test Scenario for ADAS Based on Complexity," in *SAE Technical Papers*, vol. Part F1298, no. September, sep 2017.

[122] Q. Xia, J. Duan, F. Gao, Q. Hu, and Y. He, "Test Scenario Design for Intelligent Driving System Ensuring Coverage and Effectiveness," *International Journal of Automotive Technology*, vol. 19, no. 4, pp. 751–758, aug 2018.

[123] J. Duan, F. Gao, and Y. He, "Test Scenario Generation and Optimization Technology for Intelligent Driving Systems," *IEEE Intelligent Transportation Systems Magazine*, p. 1, 2020.

[124] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 132–142.

[125] J. Wang, C. Zhang, Y. Liu, and Q. Zhang, "Traffic sensory data classification by quantifying scenario complexity," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1543–1548.

[126] C. Zhang, Y. Liu, Q. Zhang, and L. Wang, "A graded offline evaluation framework for intelligent vehicle's cognitive ability," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 320–325.

[127] J. Bolte, A. Bar, D. Lipinski, and T. Fingscheidt, "Towards corner case detection for autonomous driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 438–445.

[128] S. Hallerbach, Y. Xia, U. Eberle, and F. Koester, "Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles," *SAE International Journal of Connected and Automated Vehicles*, vol. 1, no. 2, pp. 2018–01–1066, apr 2018.

[129] R.-D. Reiss, *Approximate distributions of order statistics: with applications to nonparametric statistics*. Springer science & business media, 2012.

[130] D. R. Kuhn, R. Bryce, F. Duan, L. S. Ghandehari, Y. Lei, and R. N. Kacker, "Combinatorial testing: Theory and practice," *Advances in Computers*, vol. 99, pp. 1–66, 2015.

[131] E.-H. Choi, T. Kitamura, C. Artho, and Y. Oiwa, "Design of prioritized n-wise testing," in *IFIP International Conference on Testing Software and Systems*. Springer, 2014, pp. 186–191.

[132] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, pp. 1–29, jan 2011.

[133] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," *Data mining and knowledge discovery*, vol. 6, no. 4, pp. 393–423, 2002.

[134] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[135] J.-k. Xiao, W.-m. Li, W. Li, and X.-r. Xiao, "Optimization on black box function optimization problem," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[136] K. Meinke, F. Niu, and M. Sindhu, "Learning-based software testing: a tutorial," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2011, pp. 200–219.

[137] P. McMinn, "Search-based software testing: Past, present and future," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 153–163.

[138] C. Cotta, M. Sevaux, and K. Sörensen, *Adaptive and multilevel metaheuristics*. Springer, 2008, vol. 136.

[139] P. M. Junietz, "Microscopic and macroscopic risk metrics for the safety validation of automated driving. Ph.D. thesis," *TU Darmstadt, Darmstadt*, 2019.

[140] S. M. Mahmud, L. Ferreira, M. S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," *IATSS Research*, vol. 41, no. 4, pp. 153–163, 2017.

[141] T. Ponn, D. Fratzke, C. Gnandt, and M. Lienkamp, "Towards Certification of Autonomous Driving: Systematic Test Case Generation for a Comprehensive but Economically-Feasible Assessment of Lane Keeping Assist Algorithms," in *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, no. Vehits. SCITEPRESS - Science and Technology Publications, 2019, pp. 333–342.

[142] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, "A Survey of Algorithms for Black-Box Safety Validation," *arXiv*, may 2020.

[143] P. I. Frazier, "A Tutorial on Bayesian Optimization," *arXiv:1807.02811 [cs, math, stat]*, Jul. 2018, arXiv: 1807.02811.

[144] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[145] R. Sargent, "Optimal control," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 361–371, 2000.

[146] C. Fan, U. Mathur, S. Mitra, and M. Viswanathan, "Controller synthesis made real: Reach-avoid specifications and linear dynamics," in *Computer Aided Verification*, H. Chockler and G. Weissenbacher, Eds. Cham: Springer International Publishing, 2018, pp. 347–366.

[147] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 237–285, May 1996.

[148] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[149] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012, conference Name: IEEE Transactions on Computational Intelligence and AI in Games.

[150] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, Y. Bi, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2018, pp. 426–440.

[151] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen, "Adaptive stress testing of airborne collision avoidance systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, pp. 6C2–1–6C2–13.

[152] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8536–8542.

[153] NHTSA, "2016 fatal motor vehicle crashes: Overview," 2017.

[154] V. an der TU Dresden GmbH, "Codebook GIDAS2017," 2018.

[155] J. Bakker, H. Jeppsson, L. Hannawald, F. Spitzhüttl, A. Longton, and E. Tomasch, "Iglad-international harmonized in-depth accident data," in *Proceedings of the 25th International Conference on Enhanced Safety of Vehicles (ESV). Detroit, MI*, 2017, pp. 1–12.

[156] P. Nitsche, P. Thomas, R. Stuetz, and R. Welsh, "Pre-crash scenarios at road junctions: A clustering method for car crash data," *Accident Analysis & Prevention*, vol. 107, pp. 137–151, 2017.

[157] P. M. Bhagat, P. S. Halgaonkar, and V. M. Wadhai, "Review of clustering algorithm for categorical data," *International Journal of Engineering and Advanced Technology*, vol. 3, no. 2, 2013.

[158] Q. Chen, Y. Chen, O. Bostrom, Y. Ma, and E. Liu, "A comparison study of car-to-pedestrian and car-to-e-bike accidents: data source: the china in-depth accident study (cidas)," SAE Technical Paper, Tech. Rep., 2014.

[159] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019.

[160] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 851–862.

[161] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, dec 2014.

[162] C. Roesener, F. Fahrenkrog, A. Uhlig, and L. Eckstein, "A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour," in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 2016, pp. 1360–1365.

[163] L. Wang, F. Fahrenkrog, T. Vogt, O. Jung, and R. Kates, "Prospective safety assessment of highly automated driving functions using stochastic traffic simulation," in *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration, Detroit*, 2017.

[164] D. Zhao, H. Peng, H. Lam, S. Bao, K. Nobukawa, D. J. LeBlanc, and C. S. Pan, "Accelerated evaluation of automated vehicles in lane change scenarios," in *Dynamic Systems and Control Conference*, vol. 57243. American Society of Mechanical Engineers, 2015, p. V001T17A002.

[165] S. Zhang, H. Peng, D. Zhao, and H. E. Tseng, "Accelerated evaluation of autonomous vehicles in the lane change scenario based on subset simulation technique," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3935–3940.

[166] L. Stark, S. Obst, S. Schoenawa, and M. Düring, "Towards vision zero: Addressing white spots by accident data based adas design and evaluation," in *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE, 2019, pp. 1–6.

[167] M. Gyllenhammar, R. Johansson, F. Warg, D. Chen, H.-M. Heyn, M. Sanfridson, J. Söderberg, A. Thorsén, and S. Ursing, "Towards an operational design domain that supports the safety argumentation of an automated driving system," in *10th European Congress on Embedded Real Time Systems (ERTS 2020)*, 2020.

[168] C. W. Lee, N. Nayeer, D. E. Garcia, A. Agrawal, and B. Liu, "Identifying the operational design domain for an automated driving system through assessed risk," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1317–1322.

[169] P. Cao and L. Huang, "Application oriented testcase generation for validation of environment perception sensor in automated driving systems," SAE Technical Paper, Tech. Rep., 2018.

[170] J. Wing, "A specifier's introduction to formal methods," *Computer*, vol. 23, no. 9, pp. 8–22, 1990.

[171] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz, "Identifying unknown unknowns in the open world: Representations and policies for guided exploration," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 2124–2132.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2022.3170122, IEEE Transactions on Software Engineering

[172] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018, pp. 0210–0215.

[173] L. Li, X. Wang, K. Wang, Y. Lin, J. Xin, L. Chen, L. Xu, B. Tian, Y. Ai, J. Wang, D. Cao, Y. Liu, C. Wang, N. Zheng, and F.-Y. Wang, "Parallel testing of vehicle intelligence via virtual-real interaction," *Science Robotics*, vol. 4, no. 28, pp. 2–5, mar 2019.

[174] J. E. Stellet, T. Brade, A. Poddey, S. Jesenski, and W. Branz, "Formalisation and algorithmic approach to the automated driving validation problem," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2019-June. IEEE, jun 2019, pp. 45–51.

[175] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "Cv-hazop: Introducing test data validation for computer vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2066–2074.

[176] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[177] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence Testing for Autonomous Vehicles: A New Approach," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, jun 2016.

[178] A. Shetty, M. Yu, A. Kurzhanskiy, O. Grembek, H. Tavafoghi, and P. Varaiya, "Safety challenges for autonomous vehicles in the absence of connectivity," *Transportation Research Part C: Emerging Technologies*, vol. 128, no. June 2017, p. 103133, jul 2021.