Automotive Innovation
https://doi.org/10.1007/s42154-025-00374-8

Text2Scenario: Text-Driven Scenario Generation for Autonomous Driving Test

Xuan Cai[1] · Xuesong Bai[1,2] · Zhiyong Cui[1,3] · Danmu Xie[1] · Daocheng Fu[4] · Haiyang Yu[1,2] · Yilong Ren[1,2]

Received: 27 June 2024 / Accepted: 19 March 2025

Abstract

Autonomous driving (AD) testing constitutes a critical methodology for assessing performance benchmarks prior to product deployment. The creation of segmented scenarios within a simulated environment is acknowledged as a robust and effective strategy; however, the process of tailoring these scenarios often necessitates laborious and time-consuming manual efforts, thereby hindering the development and implementation of AD technologies. In response to this challenge, Text2Scenario is introduced, a framework that leverages a Large Language Model (LLM) to autonomously generate simulation test scenarios that closely align with user specifications, derived from their natural language inputs. Specifically, an LLM, equipped with a meticulously engineered input prompt scheme functions as a text parser for test scenario descriptions. The LLM extracts from a hierarchically organized scenario repository the components that most accurately reflect the user's preferences. Subsequently, by exploiting the precedence of scenario components, the process involves sequentially matching and linking scenario representations within a Domain Specific Language corpus, ultimately fabricating executable test scenarios. The experimental results demonstrate that such prompt engineering can meticulously extract the nuanced details of scenario elements embedded within various descriptive formats, with the majority of generated scenarios aligning closely with the user's initial expectations, allowing for the efficient and precise evaluation of diverse AD stacks void of the labor-intensive need for manual scenario configuration. Project page: https://caixxuan.github.io/Text2Scenario.GitHub.io.

Keywords Scenario generation · Large language model · Autonomous driving test · Domain specific language

Abbreviations

AD	Autonomous driving
ADAS	Advanced driver assistance system
ADS	Autonomous driving system
C-NCAP	China New Car Assessment Program
CoT	Chain of thought
DSL	Domain specific language
FS	Few-shot
LLM	Large language model
NHTSA	National Highway Traffic Safety Administration
ROS	Robot operating system
RQ	Research question
SAC	Syntax alignment checking
SC	Self-consistency
SOTIF	Safety of the Intended Functionality
SUT	System under test
SysML	Systems modeling language
T2S	Text-to-scenario
UML	Unified modeling language

Xuan Cai and Xuesong Bai have contributed equally to this work.

✉ Zhiyong Cui
zhiyongc@buaa.edu.cn

[1] State Key Laboratory of Intelligent Transportation Systems, School of Transportation Science and Technology, Beihang University, Beijing 100191, Beijing, People's Republic of China

[2] Zhongguancun Laboratory, Beijing 100191, Beijing, People's Republic of China

[3] Key Laboratory of Transport Industry of Comprehensive Transportation Theory (Nanjing Modern Multimodal Transportation Laboratory), Ministry of Transport, Nanjing 210000, Jiangsu Province, People's Republic of China

[4] Shanghai Artificial Intelligence Laboratory, Shanghai 200232, Shanghai, People's Republic of China

Published online: 06 January 2026

1 Introduction

1.1 Background

The advent of AD technology marks the onset of a transformative era, promising significant advancements in traffic safety and mobility efficiency [1, 2]. Prior to its widespread commercial deployment, the technology must undergo rigorous evaluations to validate its reliability and safety credentials [3]. Traditional vehicle safety assessments typically depend on controlled collision experiments and behavioral testing within predefined circumstances. However, Autonomous Driving Systems (ADSs), due to their intricate operational complexities and requirements for environmental adaptability, demand the creation of a diverse array of testing scenarios to comprehensively address the inherent challenges [4]. Moreover, the vast diversity of real-world traffic scenarios suggests that physical testing alone cannot fully capture all potential risk exposures [5]. Consequently, an effective approach necessitates the utilization of computational simulation technologies to generate a broad range of intricate virtual scenarios [6, 7]. These simulated environments act as arenas for rigorously evaluating and refining the effectiveness of the ADSs.

However, the prevailing methodologies for scripting AD test scenarios, which employ description languages such as Domain Specific Language (DSL), Unified Modeling Language (UML), and Systems Modeling Language (SysML), remain largely manual and require continuous maintenance [8]. This traditional approach presents considerable deficiencies and limitations. To begin with, the manual composition of DSLs for intricate, multi-actor scenarios constitutes a laborious and resource-intensive tasks, with developers committing substantial effort to guarantee scenario accuracy, consistency, and comprehensiveness. Moreover, manually crafted DSLs are prone to errors and oversights, potentially leading to insufficient test coverage or a reduction in testing rigor. For instance, scripting a basic car-following scenario may involve more than 200 lines of DSL code [9], requiring considerable time for understanding and maintenance. This process is particularly challenging for novices, presenting a steep learning curve.

An additional critical concern is the rapid advancement of AD technology, which simultaneously intensifies the demand for a continuously evolving set of test scenarios. The process of manually maintaining and updating DSLs to keep pace with these advancements is formidable and fraught with challenges, often resulting in the obsolescence and functional deficiencies of DSLs. Moreover, reliance on individual expertise and domain-specific insight during the DSL authoring process results in significant variability across scenarios crafted by different developers, thereby compromising standardization and reusability [10]. As a result, the industry urgently requires an automated DSL generation mechanism—one that comprehends developers' intentions while offering convenience, automation, standardization, and reusability.

In response to these challenges, scholarly endeavors have initiated the exploration of methodologies for the automatized generation of DSLs tailored to AD testing scenarios. LawBreaker [11] innovatively converts traffic regulations into driver-centric Signal Temporal Logic specifications and employs a fuzzy testing engine to uncover diverse modalities of rule violation by optimizing specification coverage. However, the initialization of the scenarios persists as a manual process. Contrarily, the pioneering world model, DriveDreamer-2 [12] circumvents the need for conventional DSL representations and leverages LLMs to decipher user requirements, generating detailed multi-perspective video scenarios via diffusion models. Notwithstanding, this approach does not facilitate a closed-loop interaction between the tested AD vehicle and the traffic environment, and its application is confined to the utilization in end-to-end ADS, with planning as the central objective [13].

1.2 Research Process

To reconcile the automation imperatives with empirical research advancements, an innovative framework for automated simulation testing scenario generation is introduced, referred to as Text2Scenario (T2S). Initially, complex traffic scenarios are deconstructed into fundamental components, and a hierarchical scenario repository is established to ensure complete coverage of scenario elements for the construction of the material library. Subsequent to that, a prompt-driven workflow was meticulously engineered, and a testing text parser based on an LLM was devised, enhancing the system's ability to comprehend and interpret natural language descriptions of test scenarios. Specifically, to navigate the intricacies and ambiguities inherent in natural language, the LLM employs a multi-stage in-context few-shot learning approach. This methodology allows for the direct selection of congruent components from the scenario material repository that align with the textual descriptions, thus enabling the seamless generation of accurate scenario representations.

Following this, a customized DSL corpus is constructed—encompassing controllable scenario parameters and events—to function as a reliable material repository for the targeted scenario description document. Grounded in the scenario representation, static element matching and dynamic element concatenation techniques are leveraged within a priority-based assembly architecture. This approach carefully modulates the scenario parameter values within the seed DSL document and incorporates segments for event

management, leading to the creation of a precise and standardized scenario description file. To conclude, evaluation indicators are tailored specifically to the ADS under test. These metrics facilitate the real-time evaluation of the ADS within the simulation platform, ultimately yielding comprehensive testing reports.

1.3 Contribution

The contribution lies in the three folds:

- A novel framework, Text2Scenario (T2S), is introduced for automated testing scenario generation, which is segmented into five distinctive stages and predicated on textual descriptions to enable virtual simulation testing of ADS.
- To the best of the authors' knowledge, this study represents the pioneering effort in taking advantage of the capabilities of LLM for parsing intricate natural language scenario descriptions within the realm of DSL and the subsequent generation of standardized, manipulable scenario representations.
- Harnessing a spectrum of test scenario descriptions, an extensive series of simulation experiments was conducted on the Carla platform, leading to the identification of 533 driving safety violations in ADS across 368 generated simulation scenarios. A comprehensive analysis of these test reports is instrumental in revealing technical vulnerabilities in ADS.

The structure of this paper is organized as follows. A critical review of the pertinent literature is conducted in Sect. 2. Section 3 introduces the proposed framework for generating test scenarios. The research questions and architecture of the experimental setup come under scrutiny in Sect. 4. The results of the experimental investigations are illustrated in Sect. 5, followed by a discussion on threats to validity of the work in Sect. 6. In the concluding Sect. 7, the findings of the study are synthesized, and directions for future work are proposed.

1.4 Motivation

The current landscape of AD scenario testing is encumbered by labor-intensive and time-consuming manual processes that involve translating functional scenarios detailed in natural language into explicit specific test parameters [14, 15], encoded into error-free DSL files [16]. There is an imperative requirement for an automated and controllable simulation scenario generation methodology that enhances scalability, open-endedness, standardization, and compatibility.

On the contrary, LLMs with powerful natural language generation capabilities [17], offer an automated solution, capable of rapidly generating complex and varied test scenario representations based on succinct text prompts. When juxtaposed with traditional manual and simulation methods, LLMs wield significant advantages such as:

- Efficiency: the ability to swiftly produce a substantial volume of test scenarios.
- Diversity: the generation of a broad spectrum of scenarios ensuring extensive coverage.
- Interpretability: formulation in natural language that is straightforward to comprehend and modify.
- Cost-effectiveness: obviating the need for extensive manual efforts.

In summary, LLMs hold considerable promise in supplanting the manual interpretation of human natural language, fostering standardized and adjustable scenario representations, thereby substantially expediting the testing processes for ADS.

2 Literature Review

2.1 Scenario-Based ADS Testing

Autonomous vehicles' development and deployment have intensified the need for robust validation and verification methods [18]. Among the various approaches proposed in the literature [19–21], one particularly promising avenue is scenario-based testing. By formulating scenarios that encompass diverse real-world situations, researchers aim to push the boundaries of testing methodologies.

One of the key aspects of scenario-based testing is the generation of relevant test scenarios. Ghodsi et al. [22] propose an efficient mechanism to characterize and generate testing scenarios using a state-of-the-art driving simulator. Wang et al. [23] utilize in-depth crash data involving powered two-wheelers to generate realistic testing scenarios for ADSs. Several other works have focused on the generation of challenging scenarios for AV testing. Zhou et al. [24] employ a genetic algorithm to generate scenarios that increase the probability of collisions or near-misses, which are deemed as challenging for AVs. Chen et al. [25] introduce an adversarial evaluation framework that generates lane-change scenarios to expose weaknesses in AVs' decision-making policies.

In addition to scenario generation, researchers have explored methods for efficiently searching the vast scenario space to identify critical test cases. Feng et al. [26] propose a multimodal critical-scenario search method that combines optimization techniques with supervised learning to

efficiently find scenarios that expose AV failures. In order to expand the security testing theory in connected environments, Shi et al. [27] present an integrated traffic and vehicle co-simulation framework that enables testing of connected and autonomous vehicle technologies, including vehicle-to-vehicle and vehicle-to-infrastructure communication.

Current research predominantly revolves around the identification and analysis of risk scenarios within AD through refinement and application of optimization or learning algorithms, leveraging existing datasets. Yet, this conventional approach necessitates user intervention for the initial establishment of seed scenarios, resulting in a notable absence of fully automated processes capable of producing anticipated scenario configurations derived directly from user-conceived ideas or blueprints.

2.2 LLM-Driven Scenario Generation

The advent of advanced large model technologies in recent years has indeed unlocked new possibilities in automatically generating high-quality datasets, inclusive of input scenarios pertinent to AD [28]. An emerging research endeavor involves employing LLMs for the simulation and generation of traffic scenarios with increased fidelity. The CTG++ [29] framework devised by Zhong et al. harnesses the synergy between spatiotemporal transformers and LLMs, empowering users to craft realistic and controllable traffic scenarios via intuitive natural language instructions. Moreover, Li et al. [30] have showcased the potential of LLMs in conjuring traffic scenarios from SUMO configuration files, effectively circumventing the conventional reliance on graphical editor interfaces or the labor-intensive process of manual XML file authorship. Despite these advancements, the extent of controllability within these generated scenarios remains an area deserved for further exploration and validation.

Recent research contributions reveal an intensified interest in harnessing LLMs' capabilities to assimilate traffic regulations and natural language descriptions, transforming them into explicit driving scenarios. Deng et al. [31] and Guezay et al. [32] pioneered methods wherein LLMs facilitate the automatic distillation of knowledge from traffic laws, spawning driving scenarios congruent with regulatory mandates, which subsequently serve in assessing diverse ADS software stacks. This innovation significantly bolsters automation and broadens the variety of AD test scenarios [33]. In a similar vein, Barone et al. [34] ventured into merging LLMs with automated driving scenario generation, elucidating techniques to construct and refine the driving conduct of AI-powered vehicles predicated on natural language scripts. Extending the application of LLMs beyond the automotive sphere, Cao et al. [35] proposed a novel approach to craft robot behavior trees. Their methodology leverages LLMs for designing and autonomously realizing intricate cross-domain tasks through an intelligible behavior tree architecture.

While the domain of scenario generation leveraging LLMs is experiencing rapid growth, it currently faces a shortfall in tailored research aimed at crafting scenarios that align squarely with user expectations and benefit from LLM-assisted unified DSL creation. Distinct from prevailing studies in the arena of scenario generation, the approach capitalizes on the capabilities of LLMs to produce standardized, manageable, and universally applicable scenario description files via linguistic and textual inputs. This methodology ensures compatibility across a broad spectrum of test subjects and environments, marking a significant advancement towards more versatile and user-centric scenario generation frameworks.

3 Methodology

This section delineates the proposed framework, T2S, for testing scenario generation. Commencing with the insertion of the testing language text into the LLM-empowered parser, it comprehends the user's specifications and outputs the scenario representation leveraging the hierarchical scenario repository. Following this, the DSL-based scenario generator is employed to fabricate standardized scenario files. These files are subsequently fed into the simulation environment, whereupon critical evaluative metrics are real-time monitored.

3.1 Overview

Figure 1 elucidates the T2S testing scenario generation framework advanced in this research. The T2S system transmutes testing descriptive text into simulator-ready DSL scripts across five distinct stages, anchored by the ASAM OpenScenario [16] textbook approach. The first stage necessitates listing the sextet of pivotal traffic scenario elements, in accordance with the Safety of the Intended Functionality (SOTIF)[36] tenets, which will be exhaustively expounded upon in Sect. 3.2. Notably, vehicle-to-everything communication level is exempted from this discourse, as it falls outside the scope of this paper.

The secondary stage involves the LLM decoding user-provided testing text into logical scenarios by the knowledge base of scenario descriptions, identifying the relevant hierarchical pre-configured elements. The following stage proceeds to match the scenario primitive, employing a priority-based assembling methodology to concatenate scenario descriptors into a coherent DSL script, thus converting these logical scenarios into concrete, executable scenarios.

Ultimately, this scenario descriptor file (DSL) is integrated into the simulation platform, which interprets and

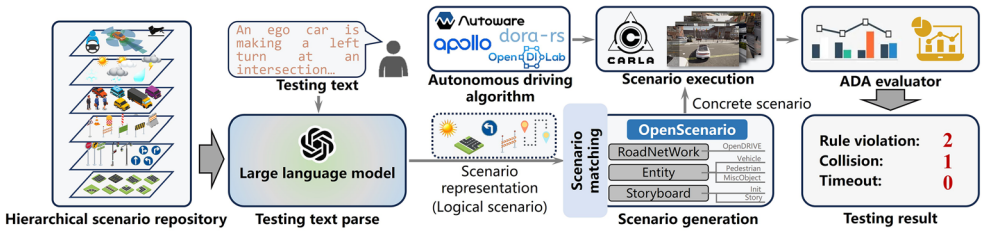

Fig. 1 Overview of the T2S testing scenario generation framework

executes the scenario. This integration supports the validation and verification of various AD stacks during the execution phase to yield testing report results.

3.2 Hierarchical Scenario Repository

Given numerous intricate components within traffic scenarios, such as ambient weather, road topologies, and unforeseen road debris, it becomes imperative to systematically categorize these elements to define and confine them to a controllable domain concisely [23]. Nonetheless, the quantification and description of many such elements present significant challenges [37], particularly regarding the subjectivity of participants' yielding behaviors or the unpredictable driving styles of individual drivers. Conventional DSLs [38, 39] necessitate precise and stringent parameter configurations to construct fully functional logical or concrete scenarios. Such rigidity in data specification inadvertently restricts the scenario's malleability, consequentially limiting the exploration of potential flaws in the system under test (SUT). To surmount this limitation, a more versatile scenario description language is advocated to enhance adaptability, enable exploratory changes, and potentially transcend the pre-established confines of stringent DSL scripts at the nascent stages.

The SOTIF-based scenario conceptualizes a hierarchical scenario framework in a systematized and canonical format, delineating interconnections between diverse components, thereby engendering logical affiliations and a tiered structure. Integration of static and dynamic elements is imperative, as the latter is contingent upon the former for contextual grounding. The communication status is intentionally discounted, given the singular emphasis on the self-driving entity. The 6-tiered edifice of hierarchical scenario depiction is assembled layer upon layer, ascending from foundational components to nuanced particulars, as depicted in Fig. 2. Leaning on the pre-crash documentation by the National Highway Traffic Safety Administration (NHTSA) [40], OpenXOntology [41], and the Traffic Safety Handbook [42], Table 1 enumerates potential elements for every component—road topology, for instance, might encompass intersections, roundabouts, and T-junctions. These elements harbor the capacity to encapsulate extensive semantic connotations, embracing the majority of typical scenario constituents encountered in the real world. Their ordered combination yields a comprehensive emulation of virtually any envisaged target scenario.

Table 1 Examples of optional elements for hierarchical scenario repository

Component	Element
Road topology	Topology: intersection, roundabout, T-junction, etc. Lanes: single lane, two-lanes, three-lanes, etc
Transportation facilities	Road markers: solid line, double solid line, broken line, etc. Traffic signs: traffic light, stop sign, speed limit sign, etc
Temporary changes	Type: cone barrel, warning sign, warning bucket, etc. Position: front, left, right, etc
Traffic participants	Type: car, truck, van, etc. Position: front, left, right, etc. Oracle: longitudinal (yield, accelerate, decelerate), lateral (keep lane, change lane), global behavior (go forward, turn)
Climate	Weather: type (sunny, rainy, snowy) and density (strong, medium, weak) Time: daytime, nighttime, morning, etc
Ego vehicle	Type: car, truck, van, etc. Position: roadside, right lane, left lane, etc. Global behavior: go forward, turn left, turn right, etc

Fig. 2 Hierarchical scenario description

- Road Topology: Establishes the foundation upon which all scenario elements rest.
- Transportation Facilities: Erected atop road topologies, these structures facilitate transport.
- Temporary Changes: Temporary modifications to road structures and transportation facilities.
- Traffic Participants: Encompasses all entities within the background traffic flow, characterized by their static attributes and dynamic behaviors.
- Climate: Represents the array of environmental factors within the scenario that influence the preceding components as well as the functionality of vehicle entities.
- Ego Vehicle: Leverages the constructed external scenario's first five tiers to influence information and actions occurring within the vehicle's interior.

3.3 Testing Text Parsing

Building on unparalleled advancements in context comprehension, textual synthesis, and image-text correspondence, LLMs like GPT-4 [43] have ascended at a trajectory potentially rivaling or surpassing human expertise. LLMs are on the cusp of supplanting human roles in knowledge interpretation, contextual alignment, and generation of experiential narratives [44] within the domain of standardized process creation. Consequently, the T2S framework explores the use of LLMs as testing text parsers, converting user-generated natural language descriptions into structured scenario representations.

(1) Prompt engineering: Following thorough training, LLMs have amassed a knowledge repository approximating human expertise. A minimal prompt can actuate these LLMs to assimilate new information and exhibit user-anticipated outputs without necessitating alterations to their intrinsic weighting. This necessitates meticulous prompt engineering to remold the functionality of LLMs. In one case study, interaction with GPT-4 by inputting a scenario description prompt is illustrated. Referenced in Fig. 3, the case study demonstrates that GPT-4 is capable of apprehending a succinct scenario narrative and identifying critical elements within. GPT-4's ability to infer underlying road topologies, such as a "T" intersection, and discern the dynamics of transportation facilities, ego vehicle behavior, and traffic participant activities from a concise description is based on prior-knowledge.

Yet, while LLMs can distill key information from natural language, their linguistic outputs are not currently suitable for direct use in constructing testing scenarios. The incomplete coverage of components in Table 1 and the introduction of uncertain elements may prevent executable scenarios or produce imprecise guidance, leading to distortions. Furthermore, the conversion of unbounded unstructured output into the hard code format is challenging.

A prompt pipeline specifically designed for testing text parsers was developed to address these issues, depicted in

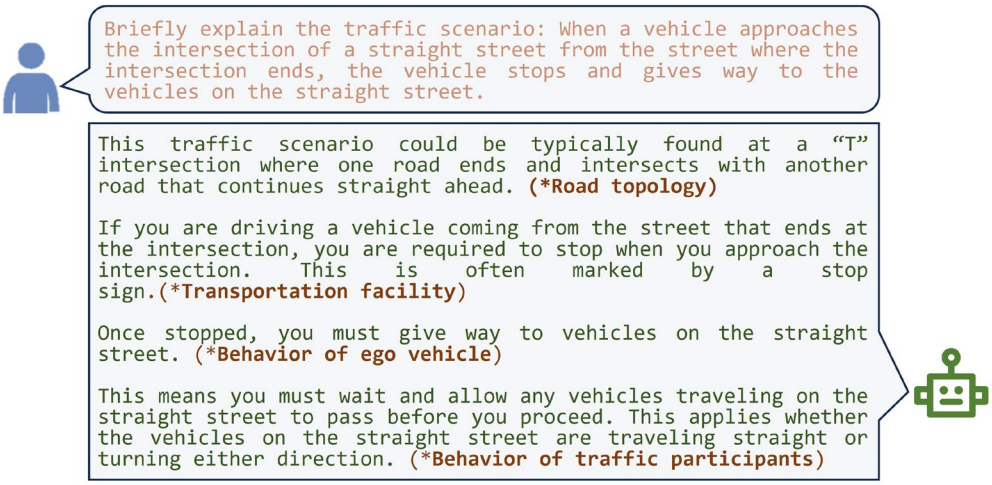

Fig. 3 A case of using prompt for interaction with GPT-4

Fig. 4. The text description undergoes sequential prompt processing to yield a scenario representation. This pipeline guides LLMs in generating structured outputs that align with DSL requirements. The Prompt Engineering process encompasses five stages: Role Setting, Few-Shot, Chain-of-Thought, Syntax Alignment Checking, and Self-Consistency. Each stage employs carefully engineered prompts to elicit stable output from the LLM. Within the first stage, Role Setting acts as the LLM's prefix prompt, positioning the LLM as an AD testing expert to refine knowledge retrieval and enhance the quality of responses, an example of which is shown in Fig. 5.

(2) Few-Shot (FS): To construct the foundational prompt that harnesses in-context few-shot learning paradigm [45], the scenario repository is embedded as a selectable knowledge base for scenario elements within the LLM. This approach enables the LLM to comprehend and reproduce structured scenario representations. This learning process is facilitated by providing a series of input–output case pairs, each consisting of scenario narrative text and its corresponding structured representation. These example pairs act as templates, providing the model with paradigms of standardized structured outputs. Ultimately, the scenario narrative that necessitates translation into a structured format is introduced. This text serves as the basic prompt trigger, which, when combined with the aforementioned input–output examples, is furnished to the LLM. This process is graphically encapsulated in Fig. 6, illustrating how the LLM synthesizes the provided information to generate structured scenario representations.

(3) Chain-of-Thought (CoT): Robust logical reasoning stands as a pillar of the "Intelligence Emergence" exhibited by LLMs [46], with the crux of reasoning hinging upon the enhancement of their thought processes. Reliance solely on an LLM's inherent knowledge repository may prove inadequate for resolving emergent and unique challenges. Frequently, the diminished certainty in an LLM's outputs can be attributed to intricate cognitive operations challenging to navigate for sophisticated tasks. For instance, when tasked with mapping the scenario "unprotected left turn for traffic vehicle" into a corresponding scenario representation, LLMs might not autonomously infer the human-favored action "yield". Instead, it may default to a "decelerate" action, erroneously conflating lower velocities as yielding.

Nonetheless, the capability of LLMs to assimilate new knowledge through systematic prompt induction—without alterations to their internal weights—is significant. By instilling fundamental thinking steps akin to those of a novice, LLMs can be trained to produce outputs that more closely align with human reasoning, thus unraveling complex tasks beyond the scope of mere few-shot prompt training.

As illustrated in Fig. 7, the given scenario text "Unprotected left turn for traffic vehicle" is dissected, with pertinent terms extracted and individually analyzed to facilitate a more deliberate contemplation by the LLM of the presented natural language stimuli. This careful analysis lays the groundwork for the LLM to formulate an appropriate

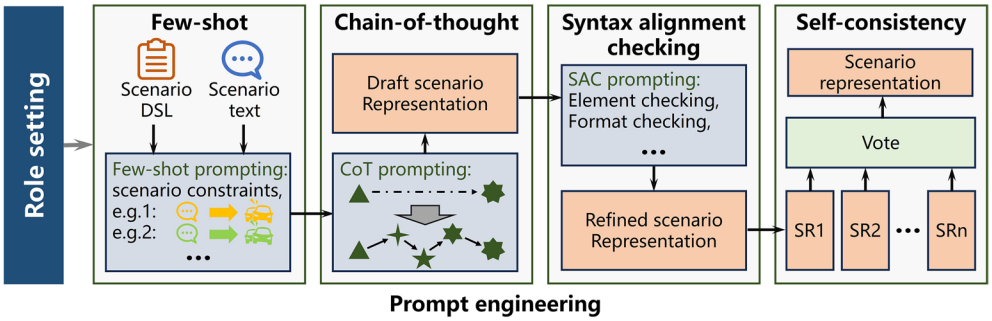

Fig. 4 Pipeline of the prompt engineering embedded in testing text parsing

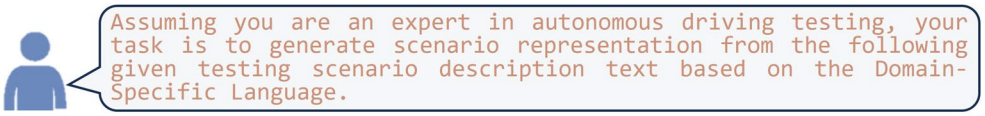

Fig. 5 Prompt of role setting of LLM

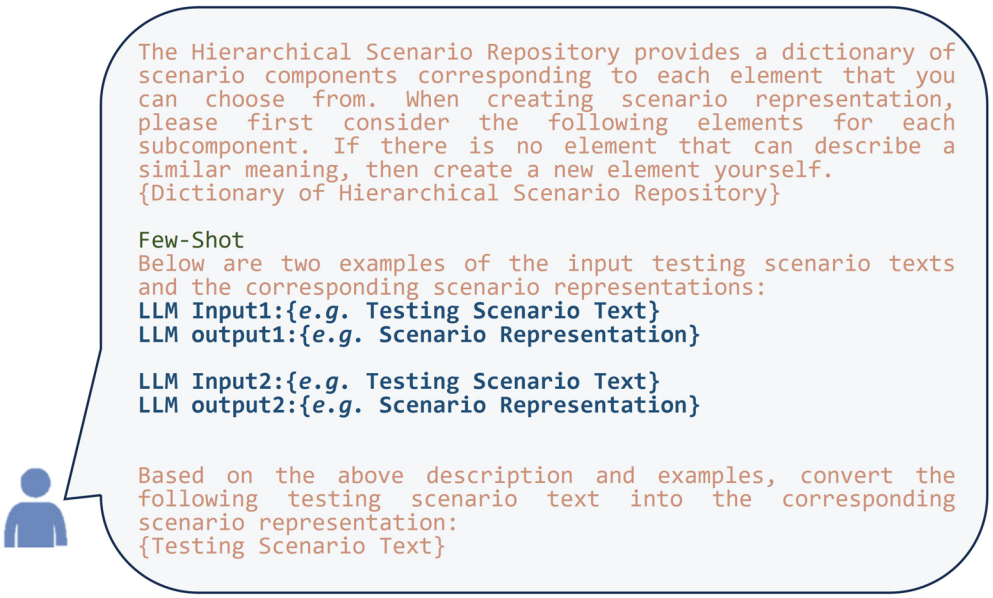

Fig. 6 Prompt of few-shot of LLM

scenario representation, hewing closely to the specified testing scenario via tailored instructional prompts.

(4) Syntax Alignment Checking (SAC): Upon the generation of a scenario representation by the LLM, it is imperative to conduct both knowledge validation and syntax harmonization on the output (Fig. 8). Knowledge validation involves cross-examining the LLM's conceptual grasp against the original scenario text to ensure congruity. Concurrently, syntax harmonization is employed to refine the output into structured data. To illustrate, an LLM may render the understanding of "straight forward" in a given scenario, while the equivalent term within the scenario repository might be cataloged as "go forward". This minor discrepancy, as minute as a singular word, could thwart hard-coded matching algorithms from accurately identifying the corresponding DSL corpus.

While precision in matching terms is crucial, it is also essential to preserve the intrinsic adaptability of the LLM. The model's unique cognitive potential may extend beyond

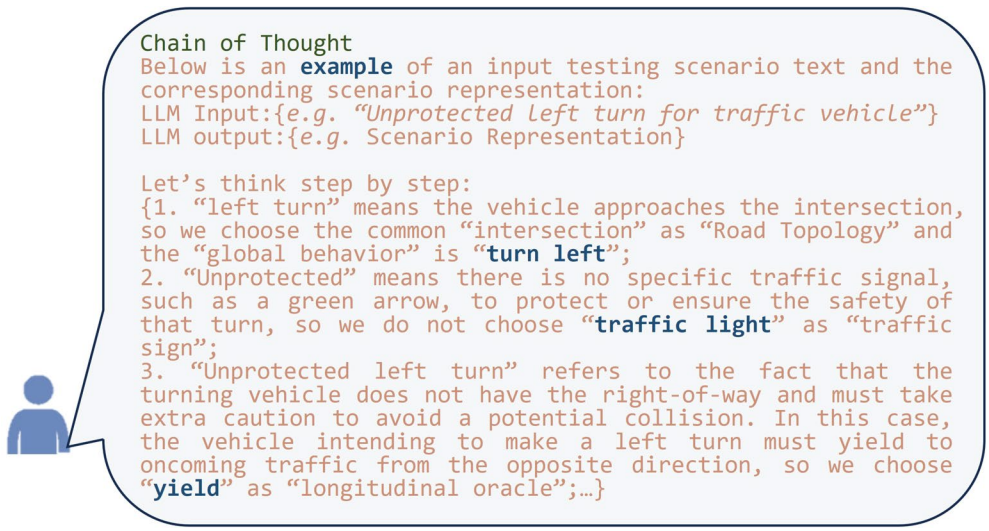

Fig. 7 Prompt of chain-of-thought of LLM

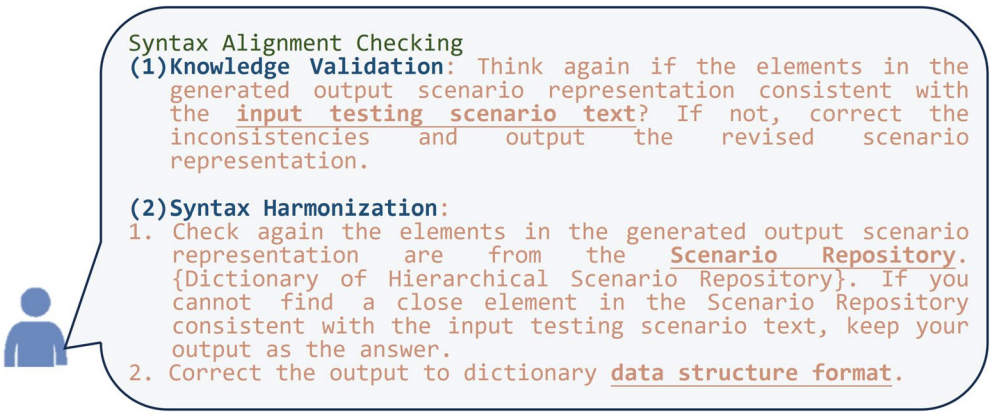

Fig. 8 Prompt of syntax alignment checking of LLM

human contemplation, permitting insight into innovative traffic elements previously unconsidered. In instances where the LLM upholds its original output due to the absence of comparable semantics within the repository, such uniqueness should be evaluated, not immediately overridden. Furthermore, the structuring of LLM output into a definitive data format is vital, sidestepping the need for laborious manual alignment during intermediary stages. It ensures that the outputs are not only accurate and reflective of the LLM's intelligence but also readily assimilable within the targeted simulation frameworks.

(5) Self-Consistency (SC): Given the inherently stochastic nature of deep learning models, the outputs produced by an LLM may exhibit a degree of randomness, leading to potential misalignments with human performance expectations. This entails a balance between response diversity and stability, modulated by hyperparameters such as temperature settings and top-k (nucleus sampling) criteria. To tackle this variability, a self-consistency strategy is employed. This approach involves the generation of several independent reasoning chains from the LLM. Diverse responses are extracted from each chain and the predominant result is determined through a consensus mechanism (or rather, majority vote [47]), drawing parallels with the deep ensemble technique [48] in deep learning to mitigate the unpredictability inherent in a singular model output. The concrete steps consist of drawing distinct inference trajectories from the LLM, followed by a marginalization process that consolidates responses to yield a final answer. As depicted in Fig. 9, the graphical illustration represents the self-consistency approach, where multiple divergent reasoning pathways converge to a dominant output labeled as the "intersection". Ten is chosen as the number of inference paths [47]. By canvassing a spectrum of potential solutions, the LLM increases the likelihood of generating accurate or significant responses to eliminate potential biases, particularly when confronted with complex tasks that challenge CoT techniques.

3.4 Domain-Specific Language-Based Scenario Generation

Acquisition of a DSL-encoded scenario description file amenable to execution by AD simulation software necessitates a systematic translation of the LLM-output scenario representation. This paper opts for the *.xosc* file format, adhering to the ASAM OpenScenario standard. The task involves meticulous link-tuning, effectively correlating the key-value pairs present within the scenario representation with their respective semantic segments as defined by OpenScenario's specification.

This study introduces a priority-based assembling architecture (Algorithm 1) tailored for static element matching and dynamic element concatenation between scenario representations and DSL corpus fragments. Commencing with the initialization of a DSL material library—the product of meticulous manual curation—this repository serves as a resource pool for populating target DSL-based files. It comprises essential event fragments such as acceleration, deceleration, lane-changing, and synchronization maneuvers amongst others [16]. The scenario representation, standardized into a dictionary-format result (e.g., .json), is derived from the LLM-powered parser as expounded upon in Sect. 3.3. An action chain is an isolated sequence of maneuvers performed by traffic entities. The objective is to delineate each standard action originating from a rudimentary scenario representation of background traffic, thereby crafting an interpretable, finely-tuned DSL control sequence. Take,

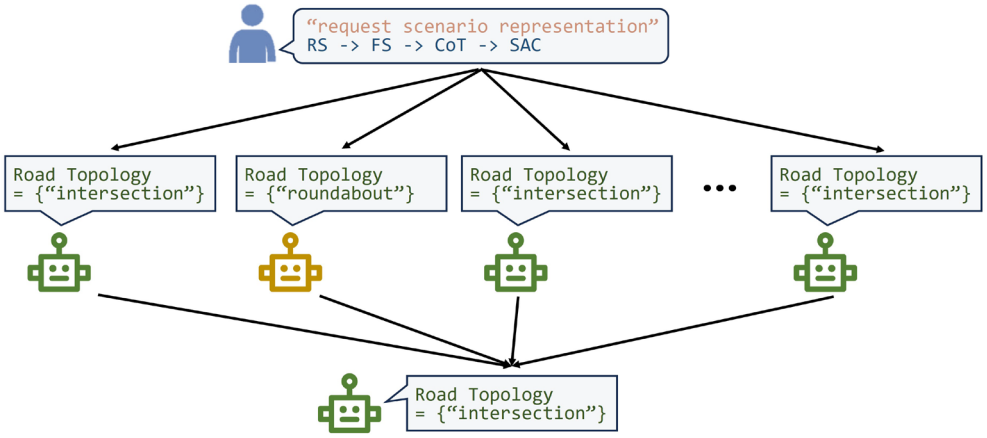

Fig. 9 A case of self-consistency of LLM

for instance, the lateral movement component within a traffic participant's scenario representation labeled as "overtaking". Such a label, while conceptually concise, encompasses a suite of discrete standard DSL actions—essentially, "change lanes—accelerate—change lanes—continue at speed"—which collectively constitute the complex maneuver traditionally recognized as "overtaking".

Algorithm 1 Priority Ranking-Based DSL Padding

Require: DSL material library $\mathcal{L}$, target DSL-based file $\mathcal{T}$, LLM scenario representation $\mathcal{R}$, action chain of traffic participants $\mathcal{A}$

▷ DSL Material Library Filling

$\mathcal{L} \leftarrow$ **Simulator Element PriorKnowledge**

$\mathcal{L} \leftarrow$ **Route Random Search**

▷ Static Element Matching

$\mathcal{T}$.**Climate**$' \xleftarrow{\mathcal{R}.Climate} \mathcal{L}$

$\mathcal{T}$.**Topology** $\xleftarrow{\mathcal{R}.Topology} \mathcal{L}$

$\mathcal{T}$.**TransportationFacilities** $\xleftarrow{\mathcal{R}.TransportationFacilities} \mathcal{L} \sim \mathcal{T}$.**Topology**

$\mathcal{T}$.**TemporaryChanges** $\xleftarrow{\mathcal{R}.TemporaryChanges} \mathcal{L} \sim \mathcal{T}$.**Topology**

$\mathcal{T}$.**EgoVehicle** $\xleftarrow{\mathcal{R}.EgoVehicle} \mathcal{L} \sim \mathcal{T}$.**Topology** & $\mathcal{T}$.**TemporaryChanges**

$\mathcal{T}$.**TrafficParticipants**.(**type**&**position_relation**) $\xleftarrow{\mathcal{R}.TrafficParticipants} \mathcal{L} \sim \mathcal{T}$.**Topology** & $\mathcal{T}$.**EgoVehicle**

▷ Dynamic Element Concatenating

$\mathbb{A} \leftarrow$ LLM powered action-chain decomposition

for a in $\mathbb{A}$ **do**

  $\mathcal{A} \xleftarrow{\mathcal{R}.TrafficParticipants.oracle} a$

end for

$\mathcal{A} \xleftarrow{\mathcal{R}.TrafficParticipants.global\ behavior} \mathcal{A}.append(\mathcal{L}.autopilot())$

$\mathcal{T} \leftarrow \mathcal{T} + \mathcal{L}(\mathcal{A})$

return $\mathcal{T}$

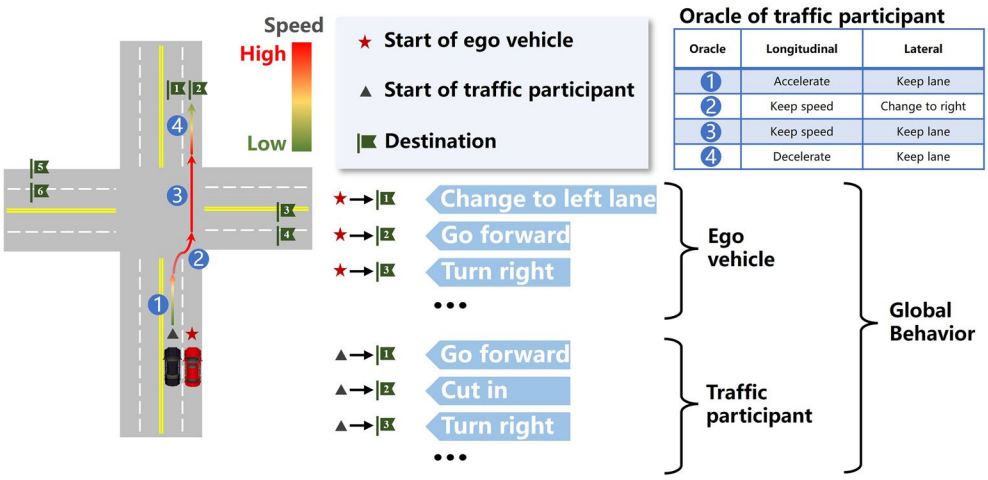

Fig. 10 Schematic diagram and labels for driving route search at intersection

Subsequent to data acquisition from the scenario simulator, the repository necessitates enrichment (refer to lines 1–2). This involves the assimilation of prior knowledge concerning simulator components, for instance, APIs for weather configuration and the selection of scenery maps. The objective is to supply route-based candidates for evaluation involving both ego vehicles and background traffic. This is realized by stochastically generating vehicular pathways—ranging from start to target points—within the set of feasible waypoints provided by the simulator, and annotating these points with precision. As illustrated in Fig. 10, consider a scenario situated within a two-lane junction. It is necessary to assign the randomly generated origin and destination coordinates with relevant labels indicating the intended interaction patterns with the ego vehicle. Each pair of route endpoints, along with their associative labels, is to be integrated into the repository, paving the way for further synergistic correlation with the key-value pairs of the scenario representation.

Subsequently, the DSL-based file designated for scenography was methodically populated with corresponding semantic units to articulate the scenario (refer to lines 3–14). The process commenced by sorting the static elements based on priorities within the hierarchical scenario repository, thereby ensuring congruity in element matching and averting redundant designations (as outlined in lines 3–8). For example, when "intersection" is delineated within ⟨road topology⟩, the ⟨road marker⟩ ought not to be signified as "broken lane", given that vehicular lane transitions at intersections are typically regulated to "solid lane" in compliance with traffic laws. Any contradictions arising in the scenario paperwork would suggest either a faltering in LLM's parsing or an erratum within the scenario's linguistic text, meriting manual scrutiny for clarification. Proceeding from lines 3 to 8, the protocol harnesses the precedence of the static hierarchy; materials are selectively retrieved from the library and embedded into the DSL-based file as delineated by the scenario narrative. The higher the hierarchical standing, the earlier its match is sought. Climate and Topology are bestowed the pinnacle of precedence. Illustrated by the expression $\mathcal{T}.\textbf{Transportation Facilities} \xleftarrow{\mathcal{R}.\textit{Transportation Facilities}} \mathcal{L} \sim \mathcal{T}.\textbf{Topology}$, it is interpreted as deriving the pertinent fragment from $\mathcal{L}$ to the key within ⟨road topology⟩ and ⟨traffic sign⟩ located in $\mathcal{R}$. The parameters of these segments are then calibrated, or a match is sought with a corresponding fragment, predicated on the value associated with the key. And the fragment elected based on $\mathcal{L}$ is subsequently transcribed to the correlating site within $\mathcal{T}$. The symbol "∼" signifies adherence to a higher hierarchical parameter within the $\mathcal{T}$'s Topology.

For the residuum encompassing dynamic elements such as **TrafficParticipants.(oracle & global behavior)**, an approach deploying action dissection and concatenation is utilized, as detailed from line 9 to 14. Figure 10 delineates an instance where the "cut in" maneuver executed by black traffic vehicles in relation to the ego vehicle can be resolved into a sequence of interconnected actions, typified by "acceleration, right-lane change, cruise, deceleration". Task labels vary from the starting point to different endpoints, facilitating scenario search matching. For instance, a traffic participant reaching endpoint 2 can be categorized into four oracle stages (indicated by the circle's number) for DSL, both longitudinally and laterally. This sequenced behavior is further deconstructed into a continuum of action chains through the use of LLM, aligning each with their respective, intricate event fragments residing in the library. Post-completion of the action chain, in scenarios where the vehicle is yet to arrive at its target terminus, the automated navigation

algorithm embedded within the action chain—as described in line 13—dictates an appendage from $\mathcal{L}$, propelling the vehicle to fulfill the comprehensive task. To encapsulate the process, the specific $\mathcal{L}(\mathcal{A})$ fragment, representing the action chain, is integrated into the designated locus within $\mathcal{T}$.

3.5 Evaluation Metrics for System Under Test

Appropriate evaluation metrics can be incorporated into the target DSL-based file to activate the simulator's monitoring system, which persistently scrutinizes the SUT behavior on a frame-by-frame basis throughout the entire execution of the scenario. These evaluative indicators encompass:

- Rule violation. This metric ascertains if the SUT contravenes traffic regulations, which entails assessments such as running stop test, running red light test, wrong lane test, and lane-keeping test. For instance, a lane invasion sensor could be configured within the Carla simulation for vigilant surveillance.
- Collision. This metric determines if the SUT has experienced any collisions with other vehicles or pedestrian objects in its vicinity. Implementing a collision detection sensor within Carla would facilitate perception and aid in recording such events.
- Time out. This metric gauges the SUT's ability to navigate to the proximity of the designated endpoint within the stipulated maximal time limit. This timeframe is adaptable and is predicated upon the length of the ADS route as well as the velocity constraint factors inherent within the scenario.

4 Simulation Experiment

This section presents an empirical validation of the T2S framework, achieved by processing diverse textual inputs and executing comprehensive simulation experiments. The implementation utilizes the ASAM OpenScenario standard format [41]. The simulations are conducted using the Carla [49] simulator, and a customized DSL file parser is integrated into the optimized Carla scenario execution environment.

4.1 Research Questions

To ascertain the efficacy of the T2S framework, this study designed three research questions (RQs) targeting key aspects of scenario generation performance. RQ1 examines the extent to which the LLM-powered testing text parsers align with the granularity of different scenario groundtruth. RQ2 investigates the fidelity with which T2S reproduces scenarios from testing description texts. RQ3 explores the effectiveness of the T2S evaluation mechanism in assessing the driving conduct of diverse SUTs.

- RQ1: How effectively does T2S's testing text parser perform at capturing the nuances across various scenario complexity levels?
- RQ2: Is T2S capable of accurately translating comprehensive testing description texts into executable scenarios?
- RQ3: Does T2S facilitate precise evaluation of the driving behaviors for various of SUTs?

4.2 Benchmark

4.2.1 Testing Text Benchmark

To ensure the scientific rigor of the testing text inputs, this study has selected three exemplary scenario description languages that are illustrative of the industry standards: (1) The pre-collision scenario descriptions outlined by the National Highway Traffic Safety Administration (NHTSA) [40]; (2) The management regulations released in the Active Safety Advanced Driver-Assistance Systems (ADAS) Test Protocol, 2024 Edition, by the China New Car Assessment Program (C-NCAP) [50] and (3) customized natural language texts for testing scenarios, meticulously crafted by testing experts in the field.

It is important to emphasize that not every test scenario provided by the NHTSA and C-NCAP is applicable to this research. Scenarios that did not pertain to the evaluation of ADS performance or that were incompatible with simulation capabilities were omitted from the study. After a careful selection process, a total of 92 test texts were retained for the analysis (NHTSA: 23; C-NCAP: 17; Customized: 52).

4.2.2 Test Scenario Benchmark

Owing to the versatile mapping of a single testing text to various locales within the diverse map topographies in Carla, a solitary test narrative is capable of spawning multiple analogous scenarios. Utilizing the 92 curated texts, this study actualized the generation of 368 test scenarios, encompassing all six intrinsic road environments within Carla, which span urban, suburban, and rural settings.

4.2.3 Systems Under Test

This study selected four distinct types of AD stacks as SUTs, that is, Autoware [51], Apollo [52], Interfuser [53], Dora-RS[1]. Autoware and Apollo are recognized as the

[1] https://github.com/dora-rs/dora.

most sophisticated and widely-adopted application-level AD systems within the industry. Interfuser, developed by OpenDILab, is an advanced end-to-end ADS based on sensor fusion[2]. Doara-RS presents a revolutionary robotic framework, infusing modernity into robotic applications; it is a prototype that hinges on a purely visual modality for its AD capability and boasts a more flexible communication framework than ROS2[3]. Note that all SUT entities are jointly simulated via the official or third-party Carla bridge. The SUTs manage the ego vehicle through various interfaces, while test scenarios are governed by decoding a DSL script.

4.2.4 LLMs Used in Text Parse

Heterogeneous LLMs: For the task of test text parsing, this study employed three principal LLMs, including Yi-34B-Chat [54], OpenAI GPT−3.5 [43], GPT-4 [43]. In alignment with the protocol delineated in Sect. 3.3 (refer to Fig. 4), the deployment of these LLMs was facilitated through their respective official APIs, ensuring systematic invocation of the LLM agents for parsing responses.

Ablation study: To elucidate the significance of each module in the parser, ablation studies were conducted, systematically omitting individual modules to assess their impact. The complete sequence of operations for the parser: Basic Prompt (BP) - Few-Shot (FS) - Chain-of-Thought (CoT) - Syntax Alignment Checking (SAC) - Self-Consistency (SC), enabled the creation of five distinct parser configurations for the purpose of ablation studies. These configurations are denoted as LLM-BP, LLM-BP-FS, LLM-BP-FS-CoT, LLM-BP-FS-CoT-SAC, and LLM-BP-FS-CoT-SAC-SC respectively.

4.3 Experiment Settings

4.3.1 RQ1: Matching Degree Across the Element-Level of Detail with the LLM-Powered Parser

To appraise the hierarchical parsing capabilities of the testing scenario parser—specifically the degree of concordance between scenario representations and the decomposed elements of the testing text—a comparative analysis between the generated representations and the established groundtruth is imperative. In exploring the generation of virtual testing scenarios, this study has examined cutting-edge methodologies recently introduced, such as RMT [55] and TARGET [31], for text-based scenario generation. RMT was deemed incongruent as a baseline methodology since it is limited to the generation of video-based open-loop test scenarios, without the provision for a closed-loop feedback model essential for virtual simulation testing. Conversely, TARGET, with its focus solely on parsing traffic rule inputs, fell short of the requisites for crafting the comprehensive virtual scenario testing scripts central to the evaluation framework.

4.3.2 RQ2: Effectiveness of T2S in Generating Scenarios—Feasibility, Accuracy, and Efficiency

To gauge T2S's impact on mitigating the manual labor involved in constructing testing scenarios, this study enlisted four ADS testing experts to juxtapose human efforts with different LLM-based implementations of T2S, thereby providing a comprehensive assessment of their feasibility, accuracy, and efficiency.

For the evaluation of the feasibility of DSL files, this study implemented a monitor to track and log any read or run-time errors that may occur in the generated OpenScenario DSL files.

To appraise the accuracy of the scenarios generated by T2S, this study recruited 15 automotive and transportation students to manually assess the congruity between the recorded output videos of the scenarios and their corresponding textual descriptions. They quantified the accuracy of T2S's scenario generation by attributing a match score ranging from 0 to 1—using a 0.1 discrete interval scale, with higher scores indicating a greater match.

Concurrently, the efficiency of T2S's rapid and precise automated testing scenario generation was evaluated by measuring the time taken for scenario construction.

4.3.3 RQ3: Compatibility of T2S Evaluator with Heterogeneous SUTs

All 4 varieties of SUTs have been seamlessly integrated to ensure compatibility with the DSL testing files. The configuration for these SUTs is restricted to designating starting and ending points, deliberately precluding human intervention in modulating any intermediate behaviors. Throughout the execution of each scenario, the performance of the SUT is attentively observed, with the objective of recording its evaluation metrics. The data accrued from these observations is then utilized to synthesize comprehensive test reports.

4.4 Evaluation Metrics

4.4.1 RQ1: Matching Degree Across the Element-Level of Detail with the LLM-Powered Parser

To quantify the LLM-powered parser's proficiency in element-level analysis, the correctness of parsing for each

[2] https://leaderboard.Carla.org/leaderboard/. Interfuser ranks first in the Carla Leaderboard.

[3] https://github.com/ros2/ros2.

constituent sub-element within the scenarios was methodically evaluated. The parser's proficiency at managing and interpreting intricate textual information is reflected in the ratio of accurately parsed elements across the entirety of the test cases. This metric serves as an indicator of the parser's overall analytical capability.

4.4.2 RQ2: Effectiveness of T2S in Generating Scenarios—Feasibility, Accuracy, and Efficiency

The capabilities of T2S and human testing experts are quantified across three primary dimensions: feasibility, accuracy, and efficiency. Concerning feasibility, the incidence of read errors and run-time errors throughout the scenarios serves as a crucial benchmark for ascertaining the executability level of T2S. The lower, the better. With respect to accuracy, this study tasked judges with appraising two critical aspects: (1) Semantic fidelity (0, 1)—this evaluation gauges the semantic coherence between the input description and the generated scenario; and (2) Driving rationality (0, 1)—this evaluation determines the extent to which the traffic participants' behavior in the generated scenario aligns with normal traffic patterns. Regarding efficiency, this study elected to utilize the metric of scenario construction time as a basis for comparison with the proficiency and pace of professional human testing experts in scenario creation.

4.4.3 RQ3: Compatibility of T2S Evaluator with Heterogeneous SUTs

Throughout the execution phase of the scenario, three critical indicators are selected to assess the competency of the AD stack: rule violation, collision, and timeout. For the timeout criterion, a threshold is established by dividing the total distance of the test route by 10% of the speed limit, which serves as the maximum allowable time for the AD stack to complete the test route.

5 Results

5.1 RQ1: Matching Degree across the Element-Level of Detail with the LLM-Powered Parser

Table 2 (specific meaning: RT-Road Topology, TF-Transportation Facilities, TC-Temporary Changes, TP-Traffic Participants, C-Climate, EV-Ego Vehicle.) delineates the parsing precision of three different LLMs for each element when a comprehensive prompt pipeline is employed, encompassing a total of 92 language descriptions within this study. The accuracy of the scenario representations produced by the LLM-based testing text parser is contrasted with the groundtruth labeled by human experts within the original scenario descriptions. This comparison evaluates their proficiency in comprehending and extracting knowledge from textual language. ChatGPT−4.0 exhibits superior language comprehension abilities relative to Yi-34B-Chat and ChatGPT−3.5, as evidenced by its closer alignment with human-level accuracy in interpreting most scenario elements, attributable to its more advanced pre-trained model. While Yi-34B-Chat outperforms ChatGPT−4.0 in parsing certain elements slightly, it maintains a generally stronger performance than ChatGPT−3.5.

Table 2 Accuracy of hierarchical scenario element parsing for different parsers

Element	Yi-34B-Chat	ChatGPT−3.5	ChatGPT−4.0
RT.⟨topology⟩	**0.96**	0.89	0.95
RT.⟨lanes⟩	0.96	0.85	**0.98**
TF.⟨road marker⟩	0.90	0.90	**0.97**
TF.⟨traffic sign⟩	0.99	0.97	**1.00**
TC.⟨type⟩	0.83	0.80	**0.95**
TC.⟨position relation⟩	0.81	0.76	**0.92**
TP.⟨type⟩	0.95	0.89	**0.98**
TP.⟨position relation⟩	0.83	0.62	**0.83**
TP.⟨longitudinal oracle⟩	0.71	0.59	**0.74**
TP.⟨lateral oracle⟩	0.72	0.53	**0.77**
TP.⟨global behavior⟩	**0.86**	0.70	0.83
C.⟨type⟩	0.98	0.91	**0.99**
C.⟨density⟩	0.87	0.68	**0.97**
C.⟨time⟩	0.90	0.86	**0.95**
EV.⟨type⟩	0.99	0.98	**0.99**
EV.⟨position⟩	**0.95**	0.90	0.95
EV.⟨global behavior⟩	0.82	0.76	**0.86**
Average	0.88	0.80	**0.92**

The best value is highlighted in bold

Within the scope of LLM analysis, explicit static expressions such as road topology, climate, and vehicle types are typically comprehended with relative ease. However, LLMs often struggle with the interpretation of complex vehicular behaviors and the nuances of multiple traffic participants' interactions, particularly when dealing with implicit expressions. For instance, the statement "*The ego vehicle encounters a vehicle cutting into its lane from a lane of static traffic*" can be misinterpreted by an LLM, which might fail to infer the implication of numerous stationary vehicles present in that lane, sometimes only generating a single vehicle in response. This issue is frequently linked to the error propagation inherent in the input–output examples used during few-shot learning, as it has been observed that LLMs tend to produce correct results when presented with similar instances. Moving forward, this research will delve into refining prompt engineering techniques to foster

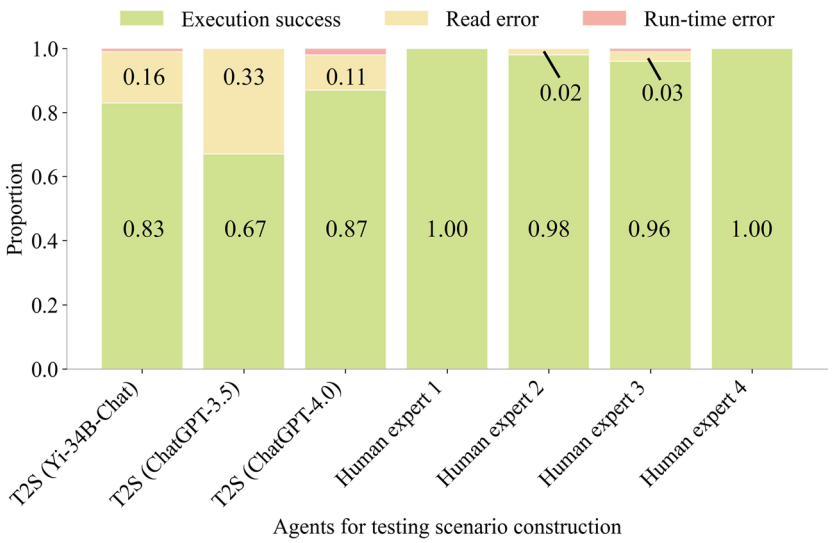

Fig. 11 Statistic proportion of DSL-file

a more encompassing comprehension of traffic components by LLMs.

In a horizontal comparative analysis among the three LLMs, ChatGPT−4.0 holds an average parsing accuracy of 92.3%, which surpasses Yi-34B-Chat and ChatGPT−3.5 by 4.2% and 11.9%, respectively, standing at 88.1% and 80.4%. This advantage is particularly noticeable in understanding dynamic behaviors within oracles, preventing significant performance degradation. Looking ahead, enhancing LLM parsing capabilities for certain concealed elements could benefit from a multimodal data analysis approach that integrates text, imagery, and video elements.

5.2 RQ2: Effectiveness of T2S in Generating Scenarios—Feasibility, Accuracy, and Efficiency

5.2.1 Feasibility

In the assessment, T2S-generated scenarios based on different LLMs have a markedly lower success rate when compared to human experts, as elucidated in Fig. 11. Human experts achieved near-perfect rates of successful script formation. This is because there is no temporal restriction on scenario construction—experts can repeatedly revise the scripts through an iterative process of reading, comprehending, authoring, debugging, testing, and modifying until they achieve success. In contrast, LLM-based T2S lacks the capability to verify the feasibility of their generated scenario representations during the construction process. This study has, however, preserved the adaptable operation space of LLM-based systems, aspiring to the principle of meticulous understanding over a rigid, hard-coded approach. Drawing insights from the human scenario construction methodology, the future research aims to improve the generation of T2S through a closed-loop feedback mechanism.

ChatGPT−4.0 maintains a significant advantage over its counterparts regarding the creation of executable scenarios, securing an 87.31% success rate, which stands considerably higher than the 5.47% and 20.21% achieved by Yi-34B-Chat and ChatGPT−3.5, respectively. The analysis uncovered that read errors typically arise when LLMs select components that fall outside the established dataset, lacking corresponding DSL fragments, which may hinder the generation of accurate DSL files. Runtime errors commonly occur due to abrupt terminations in the runtime, associated with legitimate but imperceptibly contradictory parameter configurations within the DSL text of the test scenario. Human experts, with their extensive experience in fine-tuning scenario parameters, are adept at averting such issues.

5.2.2 Accuracy

For the purpose of evaluating the accuracy of the scenario parser, this study utilized the top-performing ChatGPT−4.0. A sampling of 23 scenarios out of the 368 was randomly selected, and human evaluators were invited to rate the scenarios for semantic fidelity and driving rationality. The distribution of these scores is depicted in Fig. 12, with further details provided in the supplementary materials[4]. Acknowledging the inherent subjectivity and variability among different human judges—some being more inclined to award higher scores while others favor lower scores for the same item—this study employed the Intraclass Correlation Coefficient (ICC) bidirectional random effects model [56]. This model serves to assess the consistency or reliability of quantitative

[4] https://docs.google.com/forms/d/e/1FAIpQLSdUoDLjNLBwgTNYyCRd9KYRjg4RML9LKa-GXPeeKcCOjiSk5g/viewform?usp=sf_link.

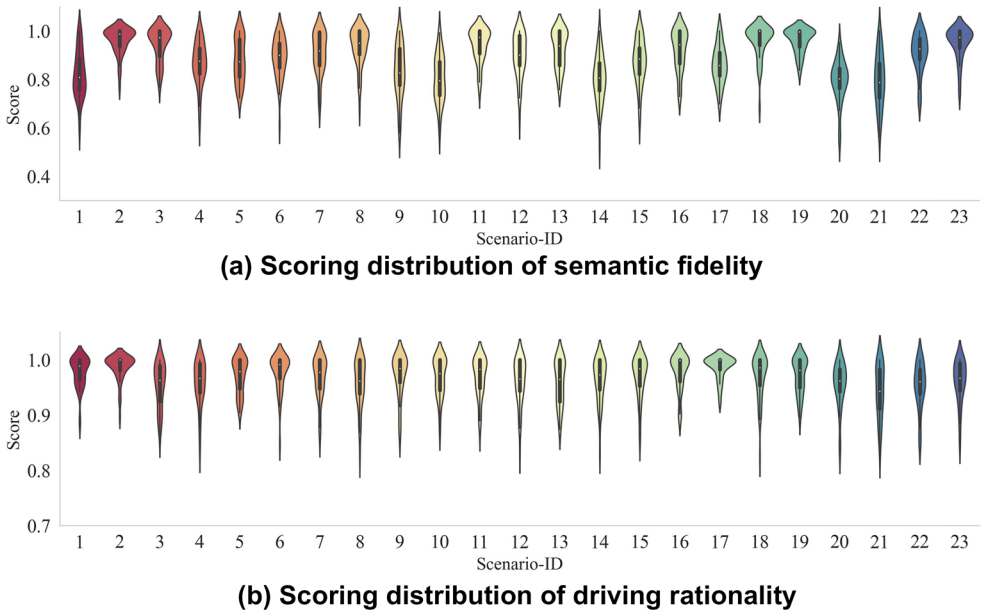

Fig. 12 Scoring distributions for 23 generated scenarios from T2S

data acquired through repeated measures, including multiple evaluators scoring the same variables. The statistical analysis revealed that the ICC value for semantic fidelity stands at 0.92, denoting superb consistency. Meanwhile, the ICC for driving rationality is 0.76, indicative of good consistency. The comparatively lower value for the latter can be ascribed to the diverse driving experiences of the evaluators, which in turn influence their perceptions and interpretations of traffic behavior. For instance, seasoned drivers might anticipate that the conduct of road users should account for interaction with other participants—a factor not encapsulated in the original DSL scenario execution file. Consequently, in certain scenarios, some traffic participants might display driving behaviors that are deemed illogical by these evaluators.

The analysis extended into the exploration of factors contributing to the lower scores assigned to certain scenarios. It appears that the deficiency in the scenario's graphical representation, particularly the inadequate depiction of water stains on the roadway (as with scenario ID 1 in subset (a)), coupled with the omission of road friction coefficients, resulted in an unrealistic portrayal of the expected loss of control. Similarly, in scenario ID 23 in subset (b), evaluators frequently noted a discrepancy between the exhibited traffic flow behaviors and standardized norms. Notwithstanding these instances, the average scores dispensed by most evaluators for both semantic fidelity and driving rationality were predominantly high, averaging 0.97 and 0.95, respectively. This affirmatively validates the capability of T2S in generating scenarios that are both semantically accurate and adherent to rational driving expectations, underscoring its utility in the intended application domain.

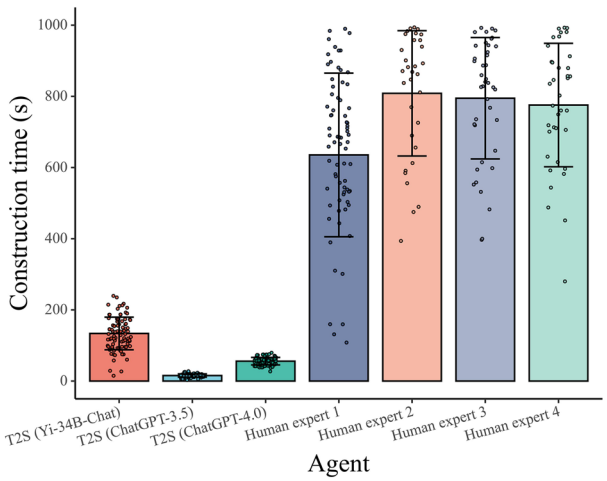

Fig. 13 Construction time with error bars

5.2.3 Efficiency

To assess the impact of the T2S framework on lessening the intensity of human labor, this study quantified the scenario construction duration for various LLM-based T2S agents and a cohort of four human experts (4 AD testing engineers, of which three have 2 years of work experience and one has more than 4 years). Demonstrated in Fig. 13 as a bar graph with error bars, the results indicate that T2S delivers a marked advantage in terms of efficiency, with construction times consistently falling below 200 s. Predominantly, time expenditure is attributed to LLM's cloud-based inference, where the ChatGPT-X series excels in response agility.

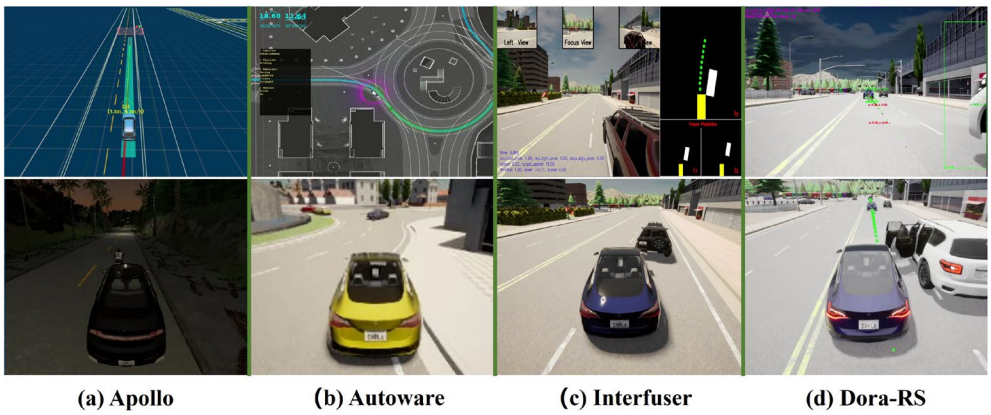

Fig. 14 Snapshots of violation scenarios for four different SUTs

Comparatively, the mean construction times for ChatGPT−3.5 and ChatGPT−4.0 stand at approximately 15 s and 55 s, respectively. The Yi-34B-Chat exhibits a longer average construction time, which is around 130 s. In stark contrast, the construction time for human experts notably exceeds that of T2S agents, with an average duration surpassing 600 s, compounding greater variability in scenario construction times. Although the efficiency of human professionals tends to improve over multiple attempts, achieving the stable processing time exhibited by T2S agents is challenging. Human participants are susceptible to grammatical and detailed errors, necessitating frequent iterative cycles of verification and debugging. In summary, the T2S framework greatly mitigates the time invested in constructing virtual scenarios, thereby diminishing the extent of human labor required.

5.3 RQ3: Compatibility of T2S Evaluator with Heterogeneous SUTs

Figure 14 delineates the snapshots of testing scenarios for four divergent SUTs—Apollo, Autoware, Interfuser, and Dora RS. Specific explanation: (a) Apollo: Chasing a two wheeled motorcycle with no lights on in the dim dusk; (b) Autoware: Violating the right solid line when entering the roundabout; (c) Interfuser: When changing lanes and overtaking, it collides with the rear end of the same lane; (d) Dora RS: The sudden opening of the door by a parked vehicle on the roadside caused a collision. The upper portion of the figure emanates from the SUT systems' developer-provided visual interfaces, while the lower segment illustrates the controlled ego vehicle within the Carla. The OpenScenario execution files, paramount for these tests, have been meticulously refined to ensure cross-compatibility with the aforementioned SUT variants. Detailing the SUTs' performance, Apollo experienced a collision, attributable to its deficient recognition capabilities in detecting two-wheeled motorcycle. Autoware, faced with an abruptly reducing target point in Carla's roundabout representation, deviated onto a high curvature trajectory, ultimately transgressing the lane boundary. Interfuser's inadequacy surfaced during a lane-change maneuver involving close-quarters interactions, where it failed to maintain a precise account of its dimensions, leading to minor abrasions. Dora-RS's perceptual mechanisms faltered, lacking the requisite training to detect out-of-distribution, such as road-side vehicles opening their doors, culminating in a failure to recognize the hazard and consequently a severe collision.

Table 3 meticulously documents the number of safety violation occurrences for each SUT within the extensive ensemble of 368 test scenarios. Apollo and Autoware have displayed commendable rule adherence and reduced collision instances, a testament to their elaborate compliance state machines and robust security frameworks that command their behavior. In stark contrast, Interfuser has been identified as the preeminent transgressor with a total of 76 rule violations, leading the cohort in this regard. Concurrently, Dora-RS held the undesirable distinction of being the most collision-prone SUT, recording 67 incidents. Analysis attributes this vulnerability to the frequent lapses in its vision-only detection system, which transmits flawed perceptual data, predisposing the system to engage in rear-end collisions or margin-centric incidents evocative of the scenarios delineated in Fig.14(d). A peculiarity was noted in the comparison between Apollo and Autoware's timeout

Table 3 Number of safety violation events for four SUTs (368 scenarios, with the maximum number of violations for each type highlighted in bold)

ADS	Rule violation	Collisions	Timeout
Apollo	31	23	49
Autoware	37	11	19
Interfuser	**76**	49	**60**
Dora-RS	61	**67**	50

Table 4 Ablation studies for the average matching accuracy (Average matching accuracy = Success rate of execution × Average accuracy of the element parsing)

LLM	BP	BP-FS	BP-FS-CoT	BP-FS-CoT-SAC	BP-FS-CoT-SAC-SC
Yi-34B-Chat	0.109	0.485	0.644	0.642	0.734
ChatGPT−3.5	0.134	0.416	0.458	0.507	0.542
ChatGPT−4.0	0.548	0.656	0.685	0.765	0.803

occurrences: Apollo presented with a surprisingly elevated rate, disparate from its projected operational capabilities. Investigations attributed this divergence to potential disparities within Guardstrikelab's Apollo-Carla-Bridge software parameters, which may sporadically lose connection, thereby precipitating signal interruption, collisions, or even inducing system inoperability. These findings have been corroborated through discussions within the open-source community forums[56]. Conclusively, under the DSL framework, T2S with Carla as its foundational simulation platform is compatible with supporting multiple SUT tests, thereby ensuring compatibility and support for parallel testing of a myriad of SUTs.

5.4 Ablation Studies

To delve deeper into the influence of individual prompt engineering modules within the LLM-based parser, this study conducted a series of ablation studies as depicted in Table 4. The analysis elucidates that, horizontally, there's a progressive enhancement in the matching accuracy of parsed outputs with the incremental inclusion of modules, particularly spotlighting the pivotal role of FS, which evidently steers the LLM toward generating outputs adherent to the pre-defined data structure, thereby curtailing compilation errors. For the Yi-34B-Chat, the CoT module markedly amplified its average matching accuracy by 0.159, starkly highlighting its effectiveness. Yet, the SAC module unexpectedly induced a marginal decrement, which could be speculated as stemming from the intrinsic randomness and indeterminacy associated with LLMs [57]. Nonetheless, this dip was adeptly mitigated by the introduction of the SC module, which, via a voting mechanism, appreciably fortified the stability and congruence of the model's outputs. From a vertical standpoint, when subjected to rigorously crafted prompt instructions, Yi-34B-Chat registered a remarkable ascent in performance, overtaking ChatGPT−3.5 with a range expanding from a −0.025 deficit on the BP to a substantial +0.192 lead when all modules were engaged (BP-FS-CoT-SAC-SC). ChatGPT−4.0 persistently exhibited a superior intellectual performance under analogous prompt conditions, notably achieving a matching accuracy index exceeding 0.4 even in the bare BP condition. Reflecting on the performance trends, ChatGPT−4.0 exhibits a markedly slower rate of degradation in capability across various prompts compared to its counterparts, substantiating its inherently robust general intelligence. In summary, each module distinctly contributes to enhancing LLMs' output proficiency: FS systematically structures the output, CoT augments the interpretive and analytical capabilities for implicit expressions, SAC rigorously scrutinizes output for syntactic and structural integrity, and SC bolsters the overall stability and reliability of LLM-generated outputs.

5.5 Discussions and Limitations

T2S streamlines the construction of scenario DSL files while preserving the fidelity and precision of scenario execution, thereby signifying a tangible reduction in manual labor. Its integration with human expertise is poised to significantly enhance simulation-based testing efficiency in ADS. Nonetheless, T2S is not devoid of challenges that necessitate strategic solutions.

- Robustness of LLM: Variations in testing scenario texts, albeit semantically identical, can precipitate disparate scenario representations due to the inherent fragility of LLM robustness—a longstanding challenge in the field of deep learning. Despite stabilization efforts through the use of SAC and SC, the resilience of LLM-based models against inconsistencies demands further refinement. Advancing the reliability of LLMs calls for additional focused training involving meticulous dataset curation and model fine-tuning.
- Scalability Constraints: While the current process incentivizes the usage of pre-existing components from the element repository, hardcoding of DSL files—though beneficial for readability, executability, and controllability—unintentionally limits the scope for probing uncharted corner cases. Future research endeavors aspire to explore innovative strategies for adaptively generating matching scenarios in the absence of pre-established element library components.
- Format Limitation: The scope of the present study confines itself to the generation of OpenScenario standard files. The specificity of the output restricts its adaptability to alternative DSL target files, thereby limiting the versatility of T2S. To overcome this bottleneck, an

[5] https://github.com/guardstrikelab/Carla_apollo_bridge/issues/159.

[6] https://github.com/dora-rs/dora/issues/660. This study was the first to receive official confirmation.

end-to-end scenario material library generation methodology—which envisions training a description file writing model through exposure to a wide range of scenario descriptions—warrants exploration. This model could independently author DSL code, shaping a new frontier in AD testing research.

6 Threats to Validity

This study discussed threats to validity from three aspects: external validity, internal validity, and construct validity, as suggested by the literature [58].

External Validity: This pertains to the applicability and generalizability of the research methods. By employing a diverse LLM approach, this study have automated the generation of test scenarios adhering to the standards of both North American NHTSA and Chinese C-NCAP, alongside expert-customized inputs. These scenarios are characterized by their cross-regional and multi-dimensional attributes and have been implemented within the simulation Carla to test ADS that follow different technological paradigms. Thanks to meticulously structured prompt engineering, this study achieved an automatic scenario generation accuracy of 80.3%, showcasing the T2S framework's adeptness in comprehending texts, matching elements, and generating scenarios. The final scenario descriptions are formatted in the internationally recognized DSL, OpenScenario, ensuring seamless adaptability to other simulation environments compatible with this standard.

Internal Validity: This concerns the direct correlation between experimental outcomes and the research methodology. Through comparative ablation studies specifically on the prompt module, this study observed that the combination of LLM and strategic prompt engineering significantly improves the success rate and accuracy of the compilation of autogenerated test scenarios. It reinforces the effectiveness of the approach in generating more reliable and applicable scenarios for ADS testing.

Construct Validity: This relates to the suitability of the ADS and simulators employed in this study. To ensure a broad representation, four ADS technologies known for their distinct operational frameworks are selected. Autoware and Apollo are modular ADS solutions prevalent in industry applications. Interfuser, notable for its high performance in the Carla leaderboard, exemplifies a multimodal end-to-end ADS indicative of potential future trends. Dora-RS, distinct for its reliance on visual processing, surpasses conventional ROS2 communication architectures in optimization. This study uses the high-fidelity simulator, Carla, as a unified platform, enabling consistent testing of the distinct ADS safety performance to ensure construct validity.

7 Conclusion and Future Direction

In conclusion, this research elucidates the design and implementation of an advanced automated scenario generation framework, Text2Scenario, which is meticulously engineered to process autonomous driving test simulations from natural language descriptions, utilizing an advanced large language model. The LLM-driven parser stringently selects pertinent scenario elements from the extensive hierarchical scenario repository that align with the provided textual descriptions. The integration of static and dynamic elements within a structured prioritization matrix subsequently facilitates the generation of executable DSL files. These files are instrumental in the real-time appraisal of ADS performance. The rigorous manual evaluation process substantiates the efficacy of the T2S approach, achieving an impressive success rate of 80.3% in generating accurate simulation scenarios across a diverse range of input texts.

This study stands at the forefront, marking an inaugural exploration into the domain of standardized DSL scenario description file generation through natural language text inputs. As the field progresses, future efforts will focus on the development of a seamless end-to-end framework for automated scenario file generation. This initiative is aimed at dismantling existing barriers associated with testing material databases. Concurrently, the synthesis of more intricate and high-risk scenarios is intended to be incorporated into the research methodology.

Appendix A: Prompt Flow for Scenario Generation

In this section, this study details the prompt flow employed to instruct the Large Language Models (LLMs) for the conversion from textual description to structured scenario representation, also known as logical scenarios, as shown in Table A1. This study utilized the publicly accessible APIs of three advanced LLMs: Yi-34B-Chat, ChatGPT−3.5, and ChatGPT−4.0, designated by their respective versions, yi-34B-v1, gpt−3.5-turbo-0613, and gpt-4-1106-preview. Post-generation, the parameters of the DSL output were meticulously adjusted by human experts to ensure an optimal level of interaction among the various traffic participants. This manual fine-tuning process was essential to accurately capture the intricate dynamics of real-world traffic scenarios in the simulations.

Table A1 Prompt flow

Role Setting. Assuming you are an expert in autonomous driving testing, your task is to generate scenario representation from the following given testing scenario description text based on the Domain-Specific Language

Hierarchical Scenario Repository. The Hierarchical Scenario Repository provides a dictionary of scenario components corresponding to each element that you can choose from. When creating scenario representation, please first consider the following elements for each subcomponent. If there is no element that can describe a similar meaning, then create a new element yourself

{*Dictionary of Hierarchical Scenario Repository*}

Few-Shot Examples. Below are two examples of the input testing scenario texts and the corresponding scenario representations:

LLM Input1: {*e.g. "Unprotected left turn for traffic vehicle"*}

LLM output1: {*e.g. Scenario Representation*}

LLM Input2: {*e.g. Testing Scenario Text*}

LLM output2: {*e.g. Scenario Representation*}

Based on the above description and examples, convert the following testing scenario text into the corresponding scenario representation:

{*Testing Scenario Text*}

Chain-of-Thought. Let's think step by step:

1. *"left turn"* means the vehicle approaches the intersection, so it should choose the common "intersection" as "Road Topology" and the "global behavior" is "turn left";
2. *"Unprotected"* means there is no specific traffic signal, such as a green arrow, to protect or ensure the safety of that turn, so it should do not choose "traffic light" as "traffic sign";
3. *"Unprotected left turn"* refers to the fact that the turning vehicle does not have the right-of-way and must take extra caution to avoid a potential collision. In this case, the vehicle intending to make a left turn must yield to oncoming traffic from the opposite direction, so it should choose "yield" as "longitudinal oracle"; ...

Syntax Alignment Checking

1. Knowledge Validation: Think again if the elements in the generated output scenario representation consistent with the input testing scenario text? If not, correct the inconsistencies and output the revised scenario representation

2. Syntax Harmonization:

(1). Check again the elements in the generated output scenario representation are from the Scenario Repository. {*Dictionary of Hierarchical Scenario Repository*}. If you cannot find a close element in the Scenario Repository consistent with the input testing scenario text, keep your output as the answer.

(2). Correct the output to dictionary data structure format

Self-consistency

Appendix B: Prompt for Scenario Description Texts

This study has meticulously cataloged a comprehensive set of 92 unique scenario description texts, as delineated in Table B1. These foundational scenarios undergo systematic variations to incorporate diverse weather conditions, road topologies, and types of traffic participants. This augmentation process multiplies the base scenarios, culminating in a total of 368 distinct scenarios that encompass a broad spectrum of driving situations.

Table B1 Scenario description texts

NTFSH

1. Control loss: Control loss without previous action. The ego-vehicle loses control due to bad RAINY conditions on the road and it must recover, coming back to its original lane
2. Traffic negotiation: Unprotected left turn at intersection with oncoming traffic. The ego-vehicle is performing an unprotected left turn at an intersection, yielding to oncoming traffic. This scenario occurs at both signalized and non-signalized junctions
3. Right turn at an intersection with crossing traffic. The ego-vehicle is performing a right turn at an intersection, yielding to crossing traffic (oncoming from the left intersection). This scenario occurs at both signalized and non-signalized junctions
4. Crossing negotiation at an unsignalized intersection. The ego-vehicle needs to negotiate with other vehicles to cross an unsignalized intersection. In this situation it is assumed that the first to enter the intersection has priority
5. Crossing traffic running a red light at an intersection. The ego-vehicle is going straight at an intersection but a crossing vehicle runs a red light, forcing the ego-vehicle to avoid the collision. This scenario occurs at both signalized and non-signalized junctions

Table B1 (continued)

6. Crossing with oncoming bicycles. The ego-vehicle needs to perform a turn at an intersection yielding to bicycles crossing from either the left or right
7. Highway. Highway merge from on-ramp. The ego-vehicle merges into moving highway traffic from a highway on-ramp
8. Highway cut-in from on-ramp. The ego-vehicle encounters a vehicle merging into its lane from a highway on-ramp. The ego-vehicle must decelerate, brake or change lane to avoid a collision
9. Static cut-in. The ego-vehicle encounters a vehicle cutting into its lane from a lane of static traffic. The ego-vehicle must decelerate, brake or change lane to avoid a collision
10. Highway exit. The ego-vehicle must cross a lane of moving traffic to exit the highway at an off-ramp
11. Yield to emergency vehicle. The ego-vehicle is approached by an emergency vehicle coming from behind. The ego-vehicle must maneuver to allow the emergency vehicle to pass
12. Obstacle avoidance. Obstacle in lane. The ego-vehicle encounters an obstacle blocking the lane and must perform a lane change into traffic moving in the same direction to avoid it. The obstacle may be a construction site, an accident or a parked vehicle
13. Door obstacle. The ego-vehicle encounters a parked vehicle opening a door into its lane and must maneuver to avoid it
14. Slow moving hazard at lane edge. The ego-vehicle encounters a slow moving hazard blocking part of the lane. The ego-vehicle must brake or maneuver next to a lane of traffic moving in the same direction to avoid it
15. Slow moving hazard at lane edge. The ego-vehicle encounters a slow moving hazard blocking part of the lane. The ego-vehicle must brake or maneuver to avoid it next to a lane of traffic moving in the opposite direction
16. Vehicle invading lane on bend. The ego-vehicle encounters an oncoming vehicles invading its lane on a bend due to an obstacle. It must brake or maneuver to the side of the road to navigate past the oncoming traffic
17. Braking and lane changing. Longitudinal control after leading vehicle's brake. The leading vehicle decelerates suddenly due to an obstacle and the ego-vehicle must perform an emergency brake or an avoidance maneuver
18. Obstacle avoidance without prior action. The ego-vehicle encounters an obstacle / unexpected entity on the road and must perform an emergency brake or an avoidance maneuver
19. Pedestrian emerging from behind parked vehicle. The ego-vehicle encounters an pedestrian emerging from behind a parked vehicle and advancing into the lane. The ego-vehicle must brake or maneuver to avoid it
20. Obstacle avoidance with prior action - pedestrian or bicycle. While performing a maneuver, the ego-vehicle encounters an obstacle in the road, either a pedestrian or a bicycle, and must perform an emergency brake or an avoidance maneuver
21. Obstacle avoidance with prior action - vehicle. While performing a maneuver, the ego-vehicle encounters a stopped vehicle in the road and must perform an emergency brake or an avoidance maneuver
22. Parking Cut-in. The ego-vehicle must slow down or brake to allow a parked vehicle exiting a parallel parking bay to cut in front
23. Parking Exit. The ego-vehicle must exit a parallel parking bay into a flow of traffic

C-NCAP

1. AEB function test. The front vehicle is stationary and the ego vehicle is driving behind it to test its AEB function
2. Vehicle crossing. The ego vehicle is driving normally, and a traffic car suddenly drives out of the side intersection, testing its obstacle avoidance ability
3. Vehicle crossing. The ego vehicle is driving normally, and a traffic vehicle suddenly emerges from a side roadway that is obscured by a large number of vehicles, testing its obstacle avoidance ability
4. Unprotected left turn. The ego vehicle wants to turn left, but the traffic car is coming from the opposite intersection
5. Passing a pedestrian. The ego vehicle is driving normally, and there is a pedestrian walking ahead on the right side of the road
6. Passing a two-wheeler. The ego vehicle is driving normally, and there is a bicycle in the opposite lane on the left side
7. Avoiding stationary vehicles in the same lane. The ego vehicle is driving normally and discovers that there is a traffic jam ahead. The ego vehicle changes lanes to the left to avoid it
8. The ego vehicle is driving normally while several traffic vehicles are parked in the right lane
9. The ego vehicle is driving normally, and there are several traffic vehicles parked on both sides of the road
10. The ego vehicle is driving along a curved road, and there is a pedestrian walking on the sidewalk on the right side
11. The ego vehicle is driving normally and suddenly detects a pedestrian crossing the road ahead
12. The ego vehicle is preparing to turn left at the intersection, but there is a traffic car parked on the route it is preparing to drive on
13. The ego vehicle is driving normally, and a traffic car in front suddenly turns right
14. The ego vehicle is driving on the left lane of a two-lane curve, with a traffic vehicle on the right side driving at a low speed. The ego vehicle overtakes it
15. The ego vehicle is preparing to change lanes to the left in the right lane of a two-lane road when suddenly a traffic vehicle drives up from the left lane
16. The ego vehicle is ready to start from a pile of stopped vehicles and change lanes to the left, but there is a motorcycle coming from the left rear

Table B1 (continued)

17. The ego vehicle is ready to start from a pile of stopped vehicles and change lanes to the left, but there is a pedestrian coming from the left rear

Customization from experts

1. An ego car was making a left turn at an intersection when a npc traffic vehicle suddenly accelerated from the right lane of ego car and changed lanes to the left, overtaking the ego car and stopping it, under very heavy rain weather conditions, with a posted speed limit of 88.5 km/h or more
2. Pedestrian crossing unexpectedly. The ego-vehicle encounters a pedestrian jaywalking or crossing the road unexpectedly, requiring immediate braking or evasive maneuvers, weather is normal cloudy
3. The left side of the highway is under repair, and there is a row of cones to guide the lane change to the right with normal sunny
4. Cyclist weaving. The ego-vehicle encounters a cyclist weaving between lanes or riding erratically, posing a collision risk
5. Debris on the road. The ego-vehicle encounters debris or objects on the road, requiring swift lane changes or braking to avoid collision, weather is extremely foggy
6. Blind intersection. The ego-vehicle approaches an intersection with limited visibility due to buildings, vegetation, or other obstructions, increasing the risk of colliding with crossing traffic
7. Unprotected left turn. The ego-vehicle needs to make an unprotected left turn across oncoming traffic, requiring precise timing and gap detection
8. Night-time driving. The ego-vehicle operates at night or in low-light conditions, where visibility is reduced, and hazards are harder to detect
9. Adverse weather conditions. The ego-vehicle operates in heavy rain, snow, fog, or other adverse weather conditions, reducing visibility and traction
10. Merging into high-speed traffic. The ego-vehicle needs to merge into a highway or high-speed traffic flow, requiring precise timing and gap detection
11. Construction zone. The ego-vehicle navigates through a construction zone with temporary lane shifts, reduced lane widths, and workers or equipment near the road
12. Stopped traffic ahead. The ego-vehicle encounters a sudden stop in traffic, requiring immediate braking to avoid rear-ending the vehicle in front
13. Aggressive driver behavior. The ego-vehicle encounters an aggressive or erratic driver tailgating, cutting in, or making sudden lane changes
14. Animal crossing. The ego-vehicle encounters an animal crossing the road, requiring immediate braking or evasive maneuvers and the weather is normal windy
15. Roundabout navigation. The ego-vehicle must navigate a multi-lane roundabout, requiring precise timing and yielding to traffic already in the roundabout
16. Narrow road with oncoming traffic. The ego-vehicle travels on a narrow road with limited clearance for oncoming traffic, requiring precise positioning and potential yielding
17. School zone navigation. The ego-vehicle navigates through a school zone with increased pedestrian activity, lower speed limits, and potential crossing guards
18. Emergency vehicle approaching. The ego-vehicle encounters an approaching emergency vehicle (police, fire, ambulance) and must yield the right-of-way
19. Toll booth navigation. The ego-vehicle must navigate through a toll booth area, potentially changing lanes or stopping to pay tolls, weather is normal cloudy
20. Parking lot maneuvers. The ego-vehicle must navigate through a crowded parking lot, with pedestrians, shopping carts, and vehicles backing out of spaces
21. Railroad crossing. The ego-vehicle approaches an active railroad crossing and must stop for a passing train or crossing gates and the weather is normal windy
22. Lane closure due to accident. The ego-vehicle encounters a lane closure due to an accident or roadwork, requiring a lane change or merging maneuver
23. Pedestrian crossing at unmarked crosswalk. The ego-vehicle encounters pedestrians crossing at an unmarked crosswalk, requiring vigilance and potential yielding
24. Bicyclist on road shoulder. The ego-vehicle encounters a bicyclist riding on the road shoulder, requiring increased clearance or lane change maneuvers and the weather is normal windy
25. Temporary traffic control devices. The ego-vehicle encounters temporary traffic control devices (cones, barricades, flaggers) due to construction or events, requiring compliance with temporary traffic patterns
26. Stopped vehicle on shoulder. The ego-vehicle encounters a stopped vehicle on the shoulder, requiring lane change maneuvers or increased clearance with little rain
27. Fallen tree or power line. The ego-vehicle encounters a fallen tree or power line blocking part of the road, requiring evasive maneuvers or lane changes with heavy wind
28. Intersection without traffic signals. The ego-vehicle approaches an intersection without traffic signals or stop signs, requiring careful navigation and yielding to other vehicles

Table B1 (continued)

29. Merge from entrance ramp. The ego-vehicle must merge from an entrance ramp onto a highway or high-speed road, requiring precise timing and gap detection
30. Exit from highway. The ego-vehicle must exit from a highway or high-speed road, requiring proper lane positioning and timing of the exit maneuver , weather is normal rainy
31. Pedestrian crossing at mid-block crosswalk. The ego-vehicle encounters pedestrians crossing at a mid-block crosswalk, requiring vigilance and potential yielding
32. Shared road with pedestrians and cyclists. The ego-vehicle travels on a road shared with pedestrians and cyclists, requiring increased awareness and caution
33. Intersection with obstructed view. The ego-vehicle approaches an intersection with limited visibility due to buildings, vegetation, or parked vehicles, increasing the risk of colliding with crossing traffic
34. Lane shift due to construction. The ego-vehicle encounters a temporary lane shift due to construction, requiring precise lane positioning and awareness of changing road patterns
35. Pedestrian crossing at signalized intersection. The ego-vehicle approaches a signalized intersection with pedestrians crossing, requiring compliancewith traffic signals and yielding to pedestrians with heavy rain
36. Stopped public transit vehicle. The ego-vehicle encounters a stopped public transit vehicle (bus, tram, train) with passengers embarking or disembarking, requiring caution and potential lane changes, weather is heavy rainy
37. Pedestrian crossing at unmarked mid-block location. The ego-vehicle encounters pedestrians crossing at an unmarked mid-block location, requiring increased vigilance and potential yielding and the weather is normal windy
38. Merge into traffic from driveway or parking lot. The ego-vehicle must merge into traffic from a driveway or parking lot, requiring precise timing and gap detection
39. Unprotected right turn on red. The ego-vehicle must make an unprotected right turn on a red light, requiring caution and yielding to pedestrians and cross traffic
40. Bicyclist riding in traffic lane. The ego-vehicle encounters a bicyclist riding in the traffic lane, requiring increased clearance or lane change maneuvers
41. Road debris from construction or accident. The ego-vehicle encounters debris or objects on the road from a construction site or accident, requiring evasive maneuvers or lane changes
42. Intersection with obstructed view due to parked vehicles. The ego-vehicle approaches an intersection with limited visibility due to parked vehicles, increasing the risk of colliding with crossing traffic, weather is heavy rain
43. Merging onto a curved highway entrance ramp. The ego-vehicle must merge onto a curved highway entrance ramp, requiring precise timing, gap detection, and lane positioning
44. Pedestrian crossing at crosswalk with limited visibility. The ego-vehicle approaches a crosswalk with limited visibility due to obstructions or lighting conditions, increasing the risk of colliding with pedestrians and the weather is normal cloudy
45. Motorcyclist lane splitting. The ego-vehicle encounters a motorcyclist lane splitting or filtering between lanes of traffic, requiring increased awareness and caution
46. Navigating through a traffic circle or roundabout. The ego-vehicle must navigate through a multi-lane traffic circle or roundabout, requiring precise timing, lane positioning, and yielding to traffic already in the circle, weather is heavy fog
47. Pedestrian crossing at intersection with obstructed view. The ego-vehicle approaches an intersection with limited visibility due to obstructions, increasing the risk of colliding with pedestrians crossing the intersection, weather is heavy fog
48. Merge from parallel parking spot. The ego-vehicle must merge into traffic from a parallel parking spot, requiring precise timing, gap detection, and awareness of surrounding traffic
49. Navigating through a toll plaza or toll booth area. The ego-vehicle must navigate through a toll plaza or toll booth area, requiring lane changes, potential stopping, and compliance with payment procedures, weather is heavy fog
50. Pedestrian crossing at mid-block location with obstructed view. The ego-vehicle encounters pedestrians crossing at a mid-block location with limited visibility due to obstructions, increasing the risk of colliding with pedestrians
51. Navigating through a parking garage or structure. The ego-vehicle must navigate through a parking garage or structure, with tight turns, potential obstacles, and pedestrians in close proximity and the weather is normal windy
52. Merging onto a highway with a short acceleration lane. The ego-vehicle must merge onto a highway with a short acceleration lane, requiring precise timing, gap detection, and acceleration to match traffic speed, weather is normal wet

Appendix C: Extensive Experiment Results

This study has provided a more detailed number of safety violation metrics for Table C1. Note that there may be multiple safety violations in a scenario.

- Running red lights (RRL): An essential aspect of traffic law adherence.
- Running stop signs (RSS): Assesses how much times the vehicle fails to stop at stop signs, a key traffic rule compliance metric.

Table C1 Number of safety violations for different SUTs

Metrics	Apollo	Autoware	Interfuser	Dora-RS
RLL	1	0	16	24
RSS	8	8	12	12
RSL	0	2	2	0
LI	4	7	10	6
CSL	4	10	15	7
WD	4	4	10	4
VRR	2	2	3	2
OR	8	4	8	6
C.	23	11	49	67
TO	49	19	60	50

- Running speed limit (RSL): Assesses how much times the vehicle exceeds speed limits, a key traffic rule compliance metric.
- Lane invasion (LI): Quantifies the numbers of lane invasions, a measure of lane-keeping accuracy.
- Crossing solid lines (CSL): Quantifies the numbers of crossing solid lines, a measure of traffic rule compliance.
- Go in a wrong direction not allowed by traffic regulations (WD): Assesses how much times the vehicle goes in a direction not allowed by traffic regulations, a key traffic rule compliance metric.
- Violation of road rights (i.e. failure to yield to pedestrians, bicycles, emergency vehicles, etc.) (VRR): Measuring the cognitive ability of vehicles in terms of road rights, a more advanced indicator for evaluating driving ability.
- Out of Road (OR): Quantifies the times the vehicle deviates from its intended roadway, indicating lane discipline.
- Collision (C.): Evaluates the times of collisions, reflecting the AV's accident avoidance capability.
- Time out (TO): Quantifies the times the vehicle fails to reach destinations in time, indicating driving capability.

Appendix D: Detailed Setup of Carla-X Co-simulation

This study utilizes Carla to facilitate traffic simulation and construct integrated simulation platforms via various bridge interfaces. This section details the methodology used to establish four SUT co-simulation platforms, collectively referred to as Carla-X.

- Apollo. Apollo-v8.0.0[7] plays the role of SUT. Carla-Apollo-bridge[8] developed by guardstrikelab is used to build communications. All parameters are configured using native settings. Apollo provides dozens of core modules, such as perception, prediction, planning, control, and human–machine interaction to achieve system level.
- Autoware. Autoware-universe[9] stack is used to test. Carla-Autoware-Bridge[10] developed by guardstrikelab is used to build communications. All parameters are configured using native settings. Autoware also adopts a modular design ethos similar to Apollo.
- Interfuser. Interfuser[11] uses multimodal fusion perception data output to plan routes. This study directly uses the pretrained weights provided by the author. The model was trained on 7 maps and 10 weather conditions. The epoch is 25, the warm-up epoch is 5, the learning rate is 0.0005, the batch size is 16, the weight decay is 0.05, the backbone learning rate is 0.0002, and the input size of multiview is $3 \times 128 \times 128$.
- Dora-RS. Dora-RS[12] is a fast and simple dataflow-oriented robotic framework with perception and control capabilities. The Yolo-v5[13], Frenet Optimal Trajectory[14] and PID[15] represent the fundamental algorithms underpinning perception, planning and control.

Acknowledgements The authors would like to appreciate the financial support of the National Key R&D Program of China, No. 2023YFB4301802-02 and No. 2022YFB4300400, the National Natural Science Foundation of China (project number: 52441202), the Beijing Natural Science Foundation (project number: L243008), the Ministry of Transport of PRC Key Laboratory of Transport Industry of Comprehensive Transportation Theory (project number: MTF2023002), and SHANDONG HI-SPEED GROUP CO,LTD [Grant No. HS2023B020].

Declarations

Conflict of interest On behalf of all the authors, the corresponding author states that there is no conflict of interest.

[7] https://github.com/ApolloAuto/apollo?tab=readme-ov-file.

[8] https://github.com/guardstrikelab/carla_apollo_bridge?tab=readme-ov-file.

[9] https://github.com/autowarefoundation/autoware.universe.

[10] https://github.com/guardstrikelab/carla_autoware_bridge.

[11] https://github.com/opendilab/InterFuser.

[12] https://github.com/dora-rs/dora.

[13] https://github.com/ultralytics/yolov5.

[14] https://github.com/erdos-project/frenet_optimal_trajectory_planner.

[15] https://github.com/Dlloydev/QuickPID.

References

1. Yang, D., Jiao, X., Jiang, K., Cao, Z.: Driving space for autonomous vehicles. Automot. Innov. **2**, 241–253 (2019)
2. Peng, L., Wang, H., Li, J.: Uncertainty evaluation of object detection algorithms for autonomous vehicles. Automot. Innov. **4**(3), 241–252 (2021)
3. Lee, D.-W., Kim, T.-L., Kwon, S.-J.: A study on the driving performance analysis for autonomous vehicles through the real-road field operational test platform. Int. J. Precis. Eng. Manuf. 1–11 (2024)
4. Gao, F., Mu, J., Han, X., Yang, Y., Zhou, J.: Performance limit evaluation strategy for automated driving systems. Automot. Innov. **5**(1), 79–90 (2022)
5. Hoss, M., Scholtes, M., Eckstein, L.: A review of testing object-based environment perception for safe automated driving. Automot. Innov. **5**(3), 223–250 (2022)
6. Chao, Q., Jin, X., Huang, H.-W., Foong, S., Yu, L.-F., Yeung, S.-K.: Force-based heterogeneous traffic simulation for autonomous vehicle testing. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 8298–8304 (2019). IEEE
7. Wang, J., Wang, X., Shen, T., Wang, Y., Li, L., Tian, Y., Yu, H., Chen, L., Xin, J., Wu, X., et al.: Parallel vision for long-tail regularization: initial results from ivfc autonomous driving testing. IEEE Trans. Intell. Veh. **7**(2), 286–299 (2022)
8. Meyer, M.-A., Sauter, L., Granrath, C., Hadj-Amor, H., Andert, J.: Simulator coupled with distributed co-simulation protocol for automated driving tests. Automot. Innov. **4**(4), 373–389 (2021)
9. Lou, G., Deng, Y., Zheng, X., Zhang, M., Zhang, T.: Testing of autonomous driving systems: where are we and where should we go? In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 31–43 (2022)
10. Li, J., Nejati, S., Sabetzadeh, M., McCallen, M.: A domain-specific language for simulation-based testing of iot edge-to-cloud solutions. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, pp. 367–378 (2022)
11. Sun, Y., Poskitt, C.M., Sun, J., Chen, Y., Yang, Z.: Lawbreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, pp. 1–12 (2022)
12. Zhao, G., Wang, X., Zhu, Z., Chen, X., Huang, G., Bao, X., Wang, X.: Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. arXiv preprint arXiv:2403.06845 (2024)
13. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al: Planning-oriented autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17853–17862 (2023)
14. Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1821–1827 (2018). IEEE
15. Zhang, P., Zhu, B., Zhao, J., Fan, T., Sun, Y.: Performance evaluation method for automated driving system in logical scenario. Automot. Innov. **5**(3), 299–310 (2022)
16. ASAM: ASAM OpenSCENARIO: User Guide (2021). https://www.asam.net/index.php?eID=dumpFile&t=f &f=4092 &
17. Xu, F.F., Alon, U., Neubig, G., Hellendoorn, V.J.: A systematic evaluation of large language models of code. In: Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, pp. 1–10 (2022)
18. Pathrudkar, S., Venkataraman, S., Kanade, D., Ajayan, A., Gupta, P., Khatib, S., Indla, V.S., Mukherjee, S.: Safr-av: Safety analysis of autonomous vehicles using real world data–an end-to-end solution for real world data driven scenario-based testing for pre-certification of av stacks. arXiv preprint arXiv:2302.14601 (2023)
19. Aparow, V.R., Hong, C.J., Weun, N.Y., Huei, C.C., Yen, T.K., Hong, L.C., Hang, C.Y., Yi, T.X., Wen, K.K.: Scenario based simulation testing of autonomous vehicle using malaysian road. In: 2021 5th International Conference on Vision, Image and Signal Processing (ICVISP), pp. 33–38 (2021). IEEE
20. Koopman, P.: Challenges in autonomous vehicle validation: Keynote presentation abstract. In: Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles, pp. 3–3 (2017)
21. Indaheng, F., Kim, E., Viswanadha, K., Shenoy, J., Kim, J., Fremont, D.J., Seshia, S.A.: A scenario-based platform for testing autonomous vehicle behavior prediction models in simulation. arXiv preprint arXiv:2110.14870 (2021)
22. Ghodsi, Z., Hari, S.K.S., Frosio, I., Tsai, T., Troccoli, A., Keckler, S.W., Garg, S., Anandkumar, A.: Generating and characterizing scenarios for safety testing of autonomous vehicles. In: 2021 IEEE Intelligent Vehicles Symposium (IV), pp. 157–164 (2021). IEEE
23. Wang, X., Peng, Y., Xu, T., Xu, Q., Wu, X., Xiang, G., Yi, S., Wang, H.: Autonomous driving testing scenario generation based on in-depth vehicle-to-powered two-wheeler crash data in china. Acc. Anal. Prevent. **176**, 106812 (2022)
24. Zhou, R., Liu, Y., Zhang, K., Yang, O.: Genetic algorithm-based challenging scenarios generation for autonomous vehicle testing. IEEE J. Radio Frequency Identif. **6**, 928–933 (2022)
25. Chen, B., Chen, X., Wu, Q., Li, L.: Adversarial evaluation of autonomous vehicles in lane-change scenarios. IEEE Trans. Intell. Transp. Syst. **23**(8), 10333–10342 (2021)
26. Feng, T., Liu, L., Xing, X., Chen, J.: Multimodal critical-scenarios search method for test of autonomous vehicles. J. Intell. Connected Veh. **5**(3), 167–176 (2022)
27. Shi, Y., Liu, Z., Wang, Z., Ye, J., Tong, W., Liu, Z.: An integrated traffic and vehicle co-simulation testing framework for connected and autonomous vehicles. IEEE Intell. Transp. Syst. Mag. **14**(6), 26–40 (2022)
28. Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al.: The rise and potential of large language model based agents: A survey. arXiv preprint arXiv:2309.07864 (2023)
29. Zhong, Z., Rempe, D., Chen, Y., Ivanovic, B., Cao, Y., Xu, D., Pavone, M., Ray, B.: Language-guided traffic simulation via scene-level diffusion. In: Conference on Robot Learning, pp. 144–177 (2023). PMLR
30. Li, Q., Peng, Z.M., Feng, L., Liu, Z., Duan, C., Mo, W., Zhou, B.: Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. Adv. Neural Inf. Process. Syst. **36** (2024)
31. Deng, Y., Yao, J., Tu, Z., Zheng, X., Zhang, M., Zhang, T.: Target: Traffic rule-based test generation for autonomous driving systems. arXiv preprint arXiv:2305.06018 (2023)
32. Güzay, Ç., Özdemir, E., Kara, Y.: A generative ai-driven application: Use of large language models for traffic scenario generation. In: 2023 14th International Conference on Electrical and Electronics Engineering (ELECO), pp. 1–6 (2023). IEEE
33. Lykov, A., Tsetserukou, D.: Llm-brain: Ai-driven fast generation of robot behaviour tree based on large language model. arXiv preprint arXiv:2305.19352 (2023)
34. Miceli-Barone, A.V., Lascarides, A., Innes, C.: Dialogue-based generation of self-driving simulation scenarios using large language models. arXiv preprint arXiv:2310.17372 (2023)
35. Cao, Y., Lee, C.: Robot behavior-tree-based task generation with large language models. arXiv preprint arXiv:2302.12927 (2023)
36. Wu, S., Wang, H., Yu, W., Yang, K., Cao, D., Wang, F.: A new sotif scenario hierarchy and its critical test case generation based on potential risk assessment. In: 2021 IEEE 1st International

Conference on Digital Twins and Parallel Intelligence (DTPI), pp. 399–409 (2021). IEEE
37. Wen, M., Park, J., Cho, K.: A scenario generation pipeline for autonomous vehicle simulators. HCIS **10**(1), 24 (2020)
38. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: a language for scenario specification and scene generation. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 63–78 (2019)
39. Queiroz, R., Berger, T., Czarnecki, K.: Geoscenario: An open dsl for autonomous driving scenario representation. In: 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 287–294 (2019). IEEE
40. Najm, W.G., Smith, J.D., Yanagisawa, M., et al.: Pre-crash scenario typology for crash avoidance research. United States. National Highway Traffic Safety Administration, Technical report (2007)
41. ASAM: ASAM OpenXOntology (2021). https://www.asam.net/standards/asam-openxontology/
42. Texas Department of Public Safety: Texas DMV Handbook. https://driving-tests.org/texas/tx-dmv-drivers-handbook-manual/
43. OpenAI: ChatGPT: Optimizing Language Models for Dialogue (2022). https://openai.com/blog/chatgpt/
44. Laskar, M.T.R., Bari, M.S., Rahman, M., Bhuiyan, M.A.H., Joty, S., Huang, J.X.: A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. arXiv preprint arXiv:2305.18486 (2023)
45. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. Adv. Neural Inf. Process. Syst. **36** (2024)
46. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)
47. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
48. Rahaman, R., et al.: Uncertainty quantification and deep ensembles. Adv. Neural. Inf. Process. Syst. **34**, 20063–20075 (2021)
49. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on Robot Learning, pp. 1–16 (2017). PMLR
50. C-NCAP: Appendix L: ACTIVE SAFETY ADAS TEST PROTOCOL [S] (2024)
51. Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y., Azumi, T.: Autoware on board: Enabling autonomous vehicles with embedded systems. In: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), pp. 287–296 (2018). IEEE
52. ApolloAuto: Apollo (2021). https://www.apollo.auto/
53. Shao, H., Wang, L., Chen, R., Li, H., Liu, Y.: Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In: Conference on Robot Learning, pp. 726–737 (2023). PMLR
54. Young, A., Chen, B., Li, C., Huang, C., Zhang, G., Zhang, G., Li, H., Zhu, J., Chen, J., Chang, J., et al.: Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652 (2024)
55. Deng, Y., Zheng, X., Zhang, T., Lou, G., Liu, H., Kim, M.: Rmt: Rule-based metamorphic testing for autonomous driving models. arXiv, 1–12 (2021)
56. Shrout, P.E., Fleiss, J.L.: Intraclass correlations: uses in assessing rater reliability. Psychol. Bull. **86**(2), 420 (1979)
57. Yin, Z., Sun, Q., Guo, Q., Wu, J., Qiu, X., Huang, X.: Do large language models know what they don't know? arXiv preprint arXiv:2305.18153 (2023)
58. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. In: Springer Berlin Heidelberg. Springer, Cham, Switzerland (2012)