scenario.center: Methods from Real-world Data to a Scenario Database

Michael Schuldes[1†], Christoph Glasmacher[1†], Lutz Eckstein[1]

Abstract—Scenario-based testing is a promising method to develop, verify and validate automated driving systems (ADS) since pure on-road testing seems inefficient for complex traffic environments. A major challenge for this approach is the provision and management of a sufficient number of scenarios to test a system. The provision, generation, and management of scenario at scale is investigated in current research. This paper presents the scenario database scenario.center to process and manage scenario data covering the needs of scenario-based testing approaches comprehensively and automatically. Thereby, requirements for such databases are described. Based on those, a four-step approach is proposed. Firstly, a common input format with defined quality requirements is defined. This is utilized for detecting events and base scenarios automatically. Furthermore, methods for searchability, evaluation of data quality and different scenario generation methods are proposed to allow a broad applicability serving different needs. For evaluation, the methodology is compared to state-of-the-art scenario databases. Finally, the application and capabilities of the database are shown by applying the methodology to the inD dataset. A public demonstration of the database interface is provided at https://scenario.center.

I. INTRODUCTION

Automated driving has great potential in increasing the efficiency and safety of traffic. To move more and more responsibility from humans to the driving function, the safety of such systems has to be established. Because of the open context [1] of urban traffic, the development, and safety validation of driving functions through testing in real-world traffic is not feasible [2]. To overcome this, scenario-based testing approaches have moved to the focus of research and industry to develop and argue for the safety of automated driving systems.

The realization of scenario-based testing has a series of challenges. To be able to define relevant test cases, an understanding of reality has to be established. This requires a large amount of diverse data from real traffic, which has to be processed and translated to scenarios. Because of the

†These authors contributed equally to this work and share first authorship.
[1]Michael Schuldes, Christoph Glasmacher, and Lutz Eckstein are with Institute for Automotive Engineering RWTH Aachen University, Aachen, Germany `firstname.lastname@ika.rwth-aachen.de`

scale, such processes have to be automated and a managing and filtering of the resulting data has to be performed. This can be achieved through a scenario database.

Therefore, this paper proposes a methodology for a scenario database including the following steps:

- Identification of scenarios in traffic data
- Computation of attributes and parameters of identified scenarios
- Storing and managing of scenario in databases
- Filtering of scenarios and providing data analysis inside the database
- Scenario generation for executable scenarios in simulation through OpenSCENARIO

The demonstrated methodology is implemented, and an application is shown. A publicly available, interactive demonstration of the scenario database is available under https://scenario.center.

II. RELATED WORK

In current research there are multiple fields approaching the topic of scenarios, as definition, processing, and storing of scenarios. For this paper, a scenario is defined as a sequence of scenes defined by actions and triggers [3].

A. Reality abstraction

Despite different definitions, the goal of a scenario is the abstraction of reality to categorize complex traffic and simplify it for simulation. For this purpose, standards like OpenSCENARIO and OpenDRIVE are created as an interface between scenarios and simulation tools [4]. To systematically describe traffic and conceptually summarize properties, scenario concepts are defined [5] serving different abstraction layers as functional, logical and concrete scenarios [6]. In [7] this abstraction is refined by subdividing logical scenarios into logical scenario classes describing the parameter declarations and logical scenario instances including distributions. Orthogonal to these deviations, further categorizations are made. [8] presents a basis for scenario concepts structuring scenario elements into 6 independent levels. These can be used to formulate an Operational Design Domain (ODD). [9] defines crash types focuses used to categorize describing accidents. These can be used to define a catalogue for scenario-based testing. [10] uses a tagging approach to describe scenarios comprehensively. [5] describes the actions and interactions of dynamic objects using a hierarchical structure based on concepts used to distiguish scenario categories in to base sceanrios/ Within [7] this concept is applied to more complex scenarios to describe recorded scenarios and generate scenarios for testing. Whereas recorded scenarios

don't have a degree of freedom, this freedom is crucial in test scenarios to allow actions of a system under test.

B. Processing of scenarios

In principle, scenarios can be created in two ways, knowledge-based and data-based. The dividing line between those principles is fluid, going from fully knowledge-based scenarios, e.g., derived from ALKS requirements [11], to fully data-based scenarios, e.g., data clustering, where even the scenario classes are derived from data, e.g., like performed in [12]. [13] demonstrates how scenarios according to the ALKS requirements are identified in data and derives concrete OpenSCENARIO files from that. [10] defines a rule based event identification from which scenarios are derived. For data-based scenario extraction, working with different data sources is essential. This can be provided through the usage of formats like the OMEGA-format [14], as hown in [7].

As an extension of identifying scenarios in data, one can generate new scenarios that fit to observed data. An example is [15], where hybrid graphs are used to estimate the parameter distribution of observed scenarios and sample new, not seen scenarios from these distributions. An extensive meta overview of the scenario generation is given in [16]. The goal of the generated scenarios is to use them for testing and evaluation in simulation. A widespread format, that is supported by various simulation tools is OpenSCENARIO [4].

C. Scenario databases

The idea of utilizing scenario databases for evaluating ADS was popularized by the Pegasus project [2]. Within the project, highway traffic was captured and described. Pegasus structures traffic into a unified scenario concept, the challenger concept [17], enabling a unified parameter space covering the pre-defined highway scenarios. But the description is limited to highway scenarios and safety relevant interaction with other traffic participants. Several other approaches to manage scenarios efficiently exist. The scenario concept of Pegasus was developed further in [18], introducing the idea of base scenarios for the motorway context.

The Sakura scenario database [19] spans from scenario identification to testing and evaluating test results. The scenarios are defined for highways. Starting from a test specification, corresponding predefined scenarios from a catalog are selected. For these scenarios, parameter distributions are extracted from data sources. These scenarios are translated to a test plan, which provides concrete scenarios to simulate. The scenarios are described using OpenSCENARIO. The simulation results can be uploaded back into the database and evaluated. The available scenarios are derived from NCAP and ISO 34502, so deriving a relation to an ODD for coverage analysis is not directly given.

The ADScene database [20] was developed out of the Moove [21] project. As part of ADScene, a set of logical highway scenarios, together with their parameters is defined.

Through a tool chain, concrete instances of these scenarios can be identified in data and aggregated to form parameter distributions for the logical scenarios. Automatic translation to simulation frameworks via formats like OpenSCENARIO is under development.

SafetyPool [22] can store functional and logical scenarios described in a specific Scenario Description Language (SDL), roughly comparable to OpenSCENARIO. Through either manually or automatically assigned tags, scenarios in the database are made searchable and the filtering to an ODD can be achieved. A parameter distribution spanning more than one scenario of the same type (e.g., a joined parameter space for two different left turn logical scenarios) can not be achieved. The export of those scenarios to simulation tools via OpenSCENARIO is supported.

Existing scenario databases either lack the support of urban scenarios or are not able to create a unified parameter space over scenarios of the same type in the database. The latter is important when arguing for coverage and completeness. Additionally, current scenario databases do not support the formulation of filters for sequences of scenarios or events. The in this work presented *scenario.center* aims at filling this gap.

III. METHODOLOGY

This paper proposes a methodological setup of a scenario database, support urban use cases and a unified parameter space for the logical scenarios. The complete process, spanning from automatic extraction of information from real-world data, over generation of logical scenario instances, querying, and generation of concrete scenarios is considered.

Before defining the structure, firstly, requirements are set for the database and underlying processes to serve different use cases for scenario-based testing:

- Scenarios have to be well-defined and understandable
- Classification and processing into scenarios has to scale
- Scenarios have to be understanble and searchable
- Scenario generation and extraction should be efficient and flexible regarding the use case
- Quality and coverage of the data should be estimated

In the following sections these requirements are used to develop methods which fulfill those requirements and enable efficient scenario-based testing.

A. Database architecture

To allow an efficient setup and orchestration of different methods to serve the given requirements, firstly the structure of the database is discussed (see Fig. 1).

To ease the usage of different data sources and to provide a fixed interface to the database logic, an input format is defined. For the proposed method, the OMEGA-format [14] is chosen. It provides converters for a diverse set of data sources and its definition follows the 6 Layer Model [8].

Based on that unified input, automated algorithms are applied (Sec. III-C) to identify sequences or events in data according to an underlying and database overarching scenario concept (Sec. III-B). The concept itself ensures a
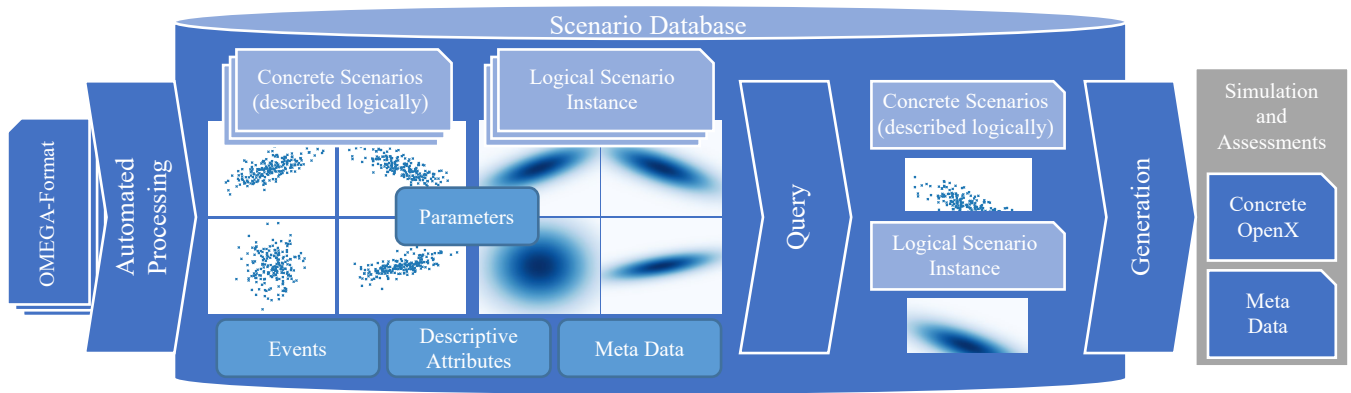
Fig. 1: Scenario-database method, spanning from data acquisition to scenario generation.

consistent representation of scenarios within the complete traffic domain. For flexible usage, categorized data as well as extracted parameters are stored both as concrete instances and logical scenario instances with distributions or value ranges. This combined storage allows the usage of real-world data as well as more sophisticated sampling methods for testing.

It might not always be of interest to work with a complete set or distributions of the whole available data. For example, only specific weather or types of intersections might be of interest. Therefore, a filtering step is included (Section III-D). From there on, to derive test cases, the logical scenario instances or the sets of concrete scenarios can be sampled. However, viable sampling methods are not in the scope of this paper.

After acquiring a concrete scenario to test or analyze, another key aspect of the scenario concept implemented in the scenario database comes into action: the ability to derive executable scenarios from each possible concrete instance of a logical scenario (Section III-E). OpenSCENARIO as an output format of the scenario, since it is supported by many industry-standard simulation tools.

B. Scenario concept

In order to define scenarios for traffic unambiguously, a comprehensive scenario concept is used for classification (see Fig. 2). The concept is thereby structured along the 6 Layer Model and utilizes the base scenario concept of [5]. Longer traffic sequences are cut to enveloping scenarios, a spatial and temporal limited scenario with an assigned ego vehicle.

For this, Layer 1 "Road Network and Traffic Guidance Objects" and Layer 4 "Dynamic Objects" are used as primary structuring elements. Within the concept abstract scenarios are described as well as attributes and parameters to concretize all the layers presented. Attributes have a descriptive character and primarily serve the annotation and discoverability of scenarios. Parameters, on the other hand, serve to generate scenarios and should therefore be orthogonal to one another or at least have well-defined relations. Both are derived from applied concepts, which serve the description of the respective concepts for the creation of base scenarios.
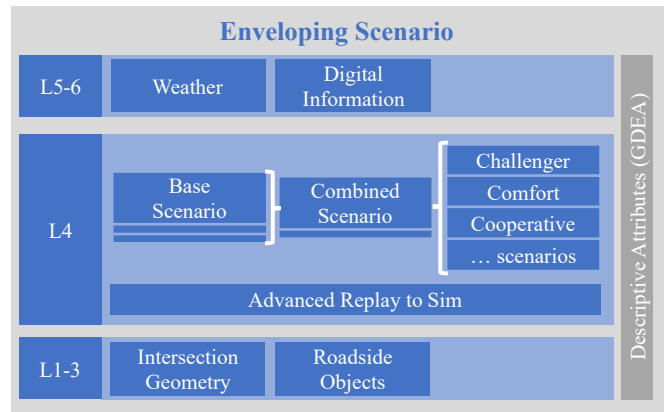

Fig. 2: Database scenario concept

In this way, the hierarchical structure ensures consistency in that the core elements are described both via parameters and attributes. Based on those, more complex combined scenarios can be created. In addition to the parametrized approach, an unparameterized approach is used to allow more detailed and real-world based scenarios.

C. Automatic extraction of scenarios

To reach an adequate coverage of the ODD, a large amount of data has to be inserted into the database. This requires a fully automated processing of data, that extracts scenarios and events. In the following the such a method is described. Each recorded drive, either real or simulated, is stored in the OMEGA-format. In a first step, the map information (Layer 1-3) is processed. Intersections and roundabouts are identified, and descriptions are derived. From a calculated intersection center, the conflict area and the angle of incoming roads are computed. Additionally, semantic relations between lanes are determined. For the description to be compact and robust, the lane sections are automatically merged and split, to, for one, respect the calculated intersection area, and ensure consistency between variations in labeling. With estimated lane and road centerlines, all objects are brought in association with lanes through Frenet coordinates similar to OpenSCENARIO. These relative coordinates are essential

for the scenarios and parameters to adapt to changing road networks.

For the extraction of dynamic information, longer driving sequences are cut into enveloping scenarios in accordance with the scenario concept. Enveloping scenarios take the perspective of an ego vehicle and split its drive into segments based on infrastructure characteristics. Within each enveloping scenario, various analyses are performed automatically. A distinction is made between event detection and the detection of base scenarios. Events assign individual attributes to specific points in time, e.g., the first point in time an object becomes visible for the ego. In contrast, base scenarios consider the semantic relationships between two road users over a time span. To detect both types, expert-defined algorithms based on common metrics are used. Furthermore, additional metrics are defined or adapted to serve the detection of e.g., interactions in urban traffic.

For efficiency, the process is designed hierarchically, so that metrics that are used multiple times only have to be calculated once and information can be detailed as needed and additional scenarios or events can be easily added (see Fig. 3).
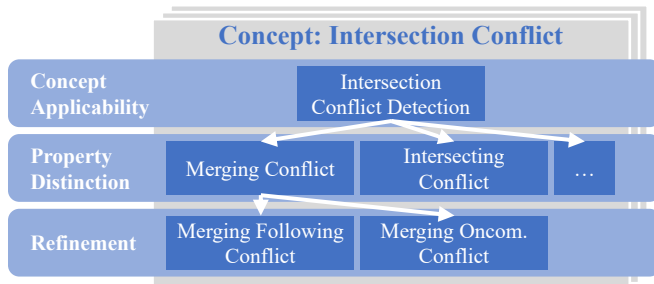

Fig. 3: Hierarchical detection of scenarios

D. Querying scenarios

To navigate through a large number scenarios efficiently relevant information must be easily accessible and dedicated to the individual use case and needed scope. Since relevance of certain inforamtion can differ between users, a flexible filtering function is required.

The flexibility is reached by filtering for specific scenarios categories, individual attributes such as metrics, parameter ranges and events. For efficient querying, this filtering options make use of the structure of processed data and the scenario concept.

The above-mentioned filtering options are commonly exposed in scenario databases. They allow the selection of specific scenarios but lack the possibility to define more complex relations and sequences of scenarios or events. To enable this, a graph-based definition of the query is developed. Such user interfaces are known from rendering and video editing software like Blender [23]. To allow an intuitive usage, users can graphically create nodes, change node features, and connect them up to form complex processes. This concept is adapted to define queries for complex scenarios and scenario sequences. To define flexible queries with the tool, it is important to define fix interfaces between nodes. Therefore, three types of nodes are defined:

- **Source nodes** specify road users and intersections
- **Filter nodes** specify filtering criteria as events and base scenarios
- **Result nodes** specify the output format and data for final or intermediate results.

With adding, configuring, and connecting nodes, the user can specify certain road users driving on specific intersections, experiencing defined events or base scenarios. Through a sequence filter node, multiple event or scenario nodes can be brought into relation, e.g., the base scenario *a* is happening right after base scenario *b*. Even complex queries can be defined with such graph-based definitions.

The challenge of this approach is the translation of this query graph into a database query, such as an SQL-statement. Each node has to be translated to a part of a query. This is achieved through the use and composition of common table expressions (CTEs). Each node can be represented through a CTE. Each node CTE has to have a fixed set of columns, so the filter nodes can rely on their presence. Each node takes the input CTEs and produces based on its filters a new output CTE. Result nodes can produce an output for the user from a CTE, e.g., displaying the set of relevant concrete scenarios or a plot of a distribution.

E. Generation of scenarios

Users of the database are likely to have different requirements and applications for scenarios. While some may only want to use them to view and analyze traffic, others may focus on re-simulating existing scenarios or sampling scenarios based on distributions. These requirements cannot be met by a single comprehensive parameterization and generation. Therefore, different generation options are offered and developed in the context of the database:

- Replay to Sim
- Advanced Replay to Sim
- Abstracted and parametrized scenarios

Within the *Replay to Sim* (RtS) approach, recorded scenarios are run according to the originally recorded scenarios. The recorded traffic events are exactly reproduced as specified in the database input, the captured data. However, this can lead to unrealistic behavior, when the behavior of the ego object in simulation deviates from the observed behavior in the source data. E.g., if an ego vehicle breaks despite not breaking in the original data, collisions with rear vehicles can occur, which have not happened in original data.

For this purpose, an *Advanced Replay to Sim* (ARtS) approach is pursued, which overwrites the original trajectory of other road users in case of such conflicts by means of a driver model [7]. Furthermore, a driver model to control the speed is implemented in the OpenSCENARIO standard, which allows deviation from the original speed profile to a necessary extent, based on time to X metrics, such as time to collision (TTC) and time headway (THW), while maintaining the original path.

In addition to the methods based on the reproduction or modification of input data, a parametrized approach to model the scenario space and provied appropriate abstractions is used. For this, the parameterization of the scenario concept is utilized. As described in Sec. III-A, input data can be described either as distributions or as concrete values. Concrete parameter values can be modified manually or can directly be translated into an OpenSCENARIO and OpenDRIVE file. Another option is to sample concrete values from a given distribution. For such sampling, a modular approach, presented in [15], is utilized. It describes relevant relations based on causal dependencies and mathematical constraints and incorporates those in the sampling logic to increase modeling performance. Using this approach offers several advantages for a database: Dependencies are described sufficiently due to causal considerations but the explosion of parameter dependencies and resulting inaccuracies for describing all multivariate relations does not occur. Furthermore, using a modular approach based on elementary parts of the scenario concepts allows definitions with reasonable amount of work. Sampling from these distributions gives concrete parameter value sets which are translated to executable scenarios.

F. Quality of database

Besides the analysis and generation of scenarios, another important part is to determine the quality of the database. For this purpose, a distinction is made into four quality and robustness criteria:

- Quality of the input data
- Well-definedness and completeness of the concept
- Quality of processing
- Coverage of scenarios in the database

Thereby, the importance and determination of these criteria is dependent on the specific use case. Following, those criteria, and the implementation of those are discussed.

A database is only as good as the input data is. Therefore, minimum quality requirements for accuracy, object representation and predefined classifications are needed for input data. This necessary quality is defined in the OMEGA format to ensure this standard within data [14]. Beside the input data, the quality of the underlying scenario concept is important to ensure the usability of the scenarios. Beside unambigiousness, understandability and the ability to create executable scenarios, a certain completeness for the concept is needed to ensure the correct classification and handling of all possible input data. For this purpose, the method for reasoning about completeness according to [24] is used. Although this is applicable for an individual use case, it cannot be done for a whole database without knowing the concrete application due to different requirements for the level of detail and described traffic. So, it is only approximable based on common use cases.

Another quality factor for the database in the processing of data. According to Sec. III-C, data is processed automatically without further labeling effort. As potential errors would have significant impact on the usability of the database, the quality of data processing within the database has to be ensured.

Therefore, it has to be guaranteed that used algorithms are robust and tested against diverse input data. Furthermore, quality checks have to be performed frequently, especially when changing input data.

The fourth measure is the coverage of real-world traffic data. This quality measure is not directly related to the systematic of the database or quality of individual input data, but to the amount and diverseness of data in the database. Giving indications for this is essential for scenario-based testing to allow a proper argumentation. To give indications about a certain coverage, a coverage approximation under input data biases can be used as described in [25].

IV. APPLICATION

To prove the applicability of the proposed methods and architectures of the scenario database, they are implemented in the scenario database *scenario.center*. As a data source, the urban trajectory dataset inD [26] is used. It contains 13.499 trajectories from four urban intersections with mixed traffic. This dataset is converted in the OMEGA-format and processed automatically by the proposed methods. According to the scenario concept (see Sec. III-B), base scenarios are derived from the given real-world data. Thereby, $7,538$ enveloping scenarios are detected. Within those, $3,566,864$ events and $59,253$ base scenarios are found. To assess the accuracy, 150 randomly chosen base scenarios are checked manually. No errors, neither classification or nor timing related ones were found. Based on those detected base scenarios, RtS, ARtS, and parametrized scenarios are generated. These scenarios are available as OpenSCENARIO and OpenDRIVE files, making them executable in a diverse set of simulation applications. Additionally, videos are created using esmini [27] and displayed in the user interface.

Besides the processing of individual scenarios, distributions are utilized within the database. Next to the generation of new scenarios as described in [15], distributions are used and visualized for comparison of different data sources. These distributions are the basis for later generation of scenarios.

A. Data analysis

Prior to generation, multiple analyses to compare and understand the underlying data can be performed with the database. This analysis step is key to have a robust simulation. The parameters and attributes of base scenarios can be explored in depth. Fig. 4 shows empirical distribution plots of the minimum THW and TTC of the base scenarios with an intersecting conflict divided by the data source. One can see that *inD Frankenburg* has fewer scenarios with a low TTC as well as fewer with a low THW compared to the other intersections. This can be explained through the difference in speed limit. The *inD Frankenburg* has a speed limit of $30km/h$, whereas the others have a speed limit of $50km/h$.

Not only the characteristics of different data sources and their influences on parameter distributions can be analyzed, but also the effects on object behavior of the scenarios themselves. As shown in Fig. 5, the intersection entrance
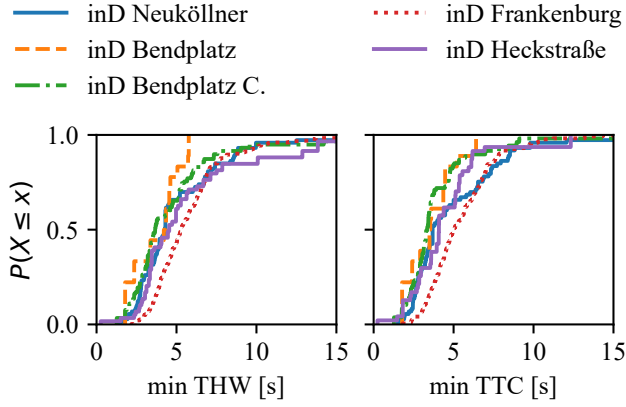
Fig. 4: Parameter distribution (ECDF) of minimum THW and TTC of the scenario concept *IntersectingConflict*.

speed of the ego vehicle tend to be lower when a conflict on the intersections occurs, compared to when no conflict occurs. A conflict thereby is a shared usage of same space within a predicated timespan. This does not mean, that approaching an intersection with higher speeds is safer, but that in most cases potential conflicts are identified in advance and the behavior is adapted to that conflict.
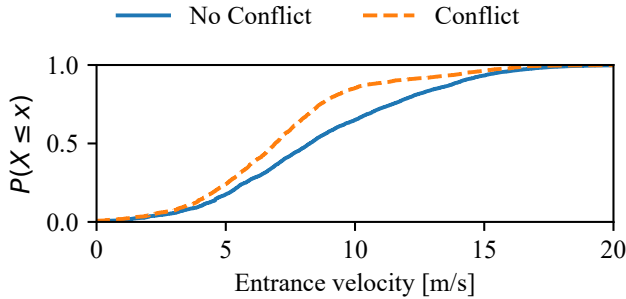

Fig. 5: Entrance velocities of ego vehicles for crossings with and without a conflict scenario.

B. Data query

As indicated in Sec. III-D, a user is only interested in a subset of scenarios at a time. Either because of a reduced ODD or because a specific situation or specific circumstances need to be inspected in more detail.

Designing an interface for filtering of metrics and attributes one to one related to the scenarios is straight forward. That is implemented with toggles for Boolean variables, (multi-) select-boxes for enumerations and sliders (and interval sliders) for scalar (interval) variables like float and times.

As described in Sec. III-D, the querying for sequences is designed in a node graph interface. Figure 6 shows an example of query formulated through the interface. Node *Ego* and *VRU* are source nodes and define desired objects, a vehicle regarded as the one experiencing the scenarios and a

road user of type vulnerable road user, meaning a pedestrian or cyclist. More detailed filtering options in each node are possible, e.g., size of the object, maximum speed or color. The filtering nodes, arbitrarily named *Following* and *Approaching* both define base scenarios where the ego vehicle moves straight through the intersection and is in following or approaching relation to the other object. Through the edge connections, it is clear, that the ego vehicle is *Ego* and the object should be that from the *VRU* node. The third filtering node *Right After* defines a time relation of both nodes. They should occur right after another. The last node defines the desired output format.
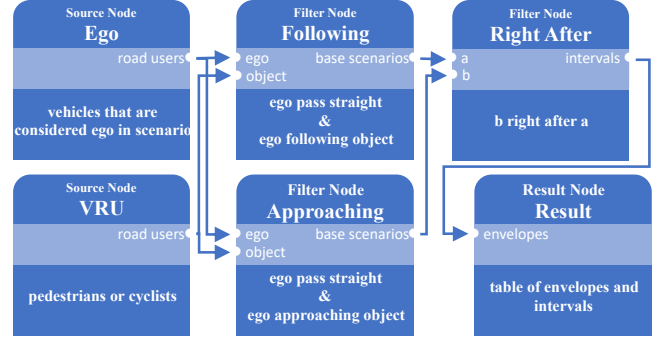

Fig. 6: Abstraction of node based user interface to interactively create queries on sequences of scenarios and events.

C. Display and generation

One result of the query defined in IV-B can be seen as a Gantt graph in Fig. 7 and Tab. I. Each bar represents a base scenario the ego vehicle experiences with different road users and the table defines these base scenarios through specifying the concept values the base scenario is defined from. Scenario sequences found via queries, random samples or broader filters are visualized in different ways to allow an easy understanding of the situation.
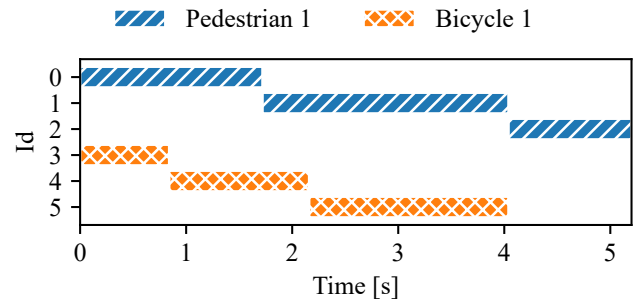

Fig. 7: Scenario timeline of an enveloping scenario as a result from a query. The base scenarios are described in Table I.

Furthermore, a video is generated to visualize the scenario in form of RtS simulated with esmini. Since the scenario is defined through OpenSCENARIO it can be simulated in CARLA, see Fig 8. Next to those visualization tools, OpenDRIVE and OpenSCENARIO files can be downloaded to use files in own simulations.

TABLE I: Scenarios in sequence of Figure 7 (*OTAC* = Object Traffic Area Change, *ROP* = Relative Object Position, *EM*= Ego Maneuver, *LS* = Longitudinal State).

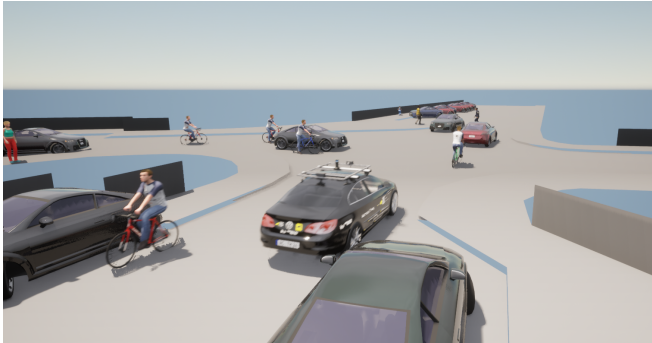| Id	OTAC [5]	ROP	EM	LS
0	-	oncoming	pass straight	-
1	crossing	oncoming	pass straight	-
2	crossing	-	-	-
3	-	same arm	pass straight	following
4	-	same arm	pass straight	approaching
5	-	same arm	pass straight	-

Fig. 8: Example image of simulation result based on XOSC and XODR acquired from the database, produced with CARLA.

Not only detailed information on individual found scenarios is of interest, but also the joined information from all found scenarios. Therefore, attribute and parameter distribution plots are available, comparable to those shown in Sec. IV-A Fig. 4, but only considering data from the selected scenarios.

D. Test execution

To check whether the generated scenarios from the database can be used to test driving functions, exemplary scenarios are taken from the query from Fig. 6. These are used to test an automatic cruise control (ACC) function in CARLA. For this, the scenarios are automatically processed to a CARLA-readable format and for a simple demonstration the TTC is taken as a measure to assess the driving function. Comparing the test scenarios with the real-world recorded constellations, lower TTCs in the simulation can be observed (see Fig. 9). This suggests, that the example ACC drives more aggressive than the observed real-world drivers.

V. DISCUSSION

This paper demonstrates the methodology of a scenario database designed for the urban use case, providing a unified parameter space for the extracted scenarios. The demonstration of the methodology through application is successfully demonstrated and accessible publicly in scenario.center. The whole process from identifying scenarios in data, over filtering to relevant scenarios, up to generating executable scenario in form of OpenSCENARIO was demonstrated.

How the developed method compares to existing approaches of scenario databases introduced in Sec. II-C is
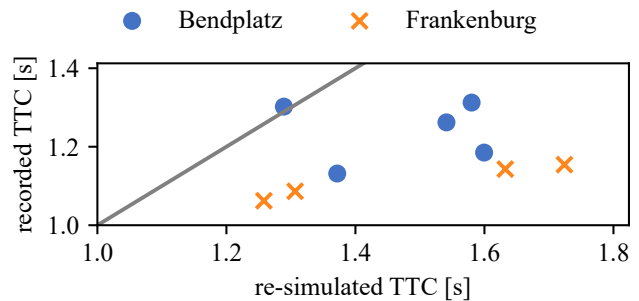

Fig. 9: Comparison of recorded and re-simulated TTCs utilizing an ACC function. The identity line is marked in gray.

TABLE II: Table comparing the different existing scenario databases.

Feature	Pegasus	Sakura	ADScene	SaftyPool™	scenario.center
highway	✓	✓	✓	✓	✓
urban	✗	✗	✗	✓	✓
automated extraction from data	✓	✓	✓	✓	✓
unified scenario concept	✓	✗	✗	✗	✓
unified parameter space	✓	✗	✗	✗	✓
sample scenario from distribution	✓	✓	✓	✓	✓
filter by sequence	✗	✗	✗	✗	✓
coverage & completeness argumentation	✗	✗	✗	✗	✓

shown in Table II. Only *SafetyPool* and *scenario.center* support the urban use case. The main differentiating factor is the underlying scenario concept. It decomposes traffic into composable building blocks, which are applicable in all traffic situations. This is key to achieve a unified parameter space per type of scenario over all data sources and a comparison between data sources. Only Pegasus demonstrated such a unified parameter space, though limited to the less complex highway setting. Such a unified parameter space also eases the argumentation for coverage and completeness, which is important for safety argumentation based on the database.

Additionally, *scenario.center* first demonstrates the application of a filtering interface for sequences of scenarios in a scenario database, and it is the only scenario database, which provides a publicly available demonstration version. The latter is important to promote the understanding of the real world capabilities and use cases of the scenario database.

Nevertheless, other frameworks provide benefits for different applications. *SafetyPool* promotes the sharing of scenarios through incentives, which is beneficial for industry-wide cooperation and strengthening synergies. The Sakura database tightly integrates testing and test results into the user interface, which demonstrates the future of other databases. Therefore, to enable one user to use all the benefits at once, interoperability between the different solutions is desired. Such a federated layer, that connects scenario databases, is the focus of the SUNRISE project [28].

Although OpenSCENARIO is used as a standard to generate scenarios it has to be stated that the applicability has some limitations. On the one hand, many assessed frameworks only support a subset of the functionality of the standard and may use different interpretations. So, targeting a broad support of tools leads to a limitation of design opportunity of scenarios. On the other hand, whereas the functionality may be sufficient for highways, further triggers would be beneficial to generate and simulate scenarios on urban intersections, especially in lateral traffic movements. Within a common database a trade-off always has to be made between usability and functionality.

Lastly, for a scenario database to sufficiently reflect a desired ODD, incorporating as much data as possible is important. However, such data is only available in limited capacity, especially, publicly available data. Moreover, the available data is of different quality and the used data formats vary, making incorporating lots of data difficult. In our work, we deal with the problem by utilizing the OMEGA-format [14] as an interface between data and scenario database. This eased the adoption of new data sources, highlighting the importance of agreed upon standards in industry and research.

VI. CONCLUSION

Within this paper, requirements and methods to create a scenario database are proposed. The developed methods are implemented in a new publicly available scenario database called *scenario.center*.

It is demonstrated how to automatically identify scenarios in data and compute corresponding parameters and attributes. Furthermore, the management of scenarios at scale and the provision of insights and filtering options to users is shown. Lastly, the generation of scenarios as input for simulation tools is described. These steps are demonstrated in an example, from acquiring data in a scenario database to assessing an example ACC.

Since there are multiple scenario databases with different capabilities and concepts, an orchestration of databases as discussed in the SUNRISE project [28] is future work.

REFERENCES

[1] International Organization for Standardization, "ISO 21448:2022 - Road vehicles - Safety of the intended functionality," 2022.

[2] H. Winner, K. Lemmer, T. Form, and J. Mazzega, "PEGASUS—first steps for the safe introduction of automated driving," in *Lecture Notes in Mobility*. Springer International Publishing, jun 2018, pp. 185–195.

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

[4] B. Engel and N. Dillmann, "Asam openx," in *Online Presentation*, vol. 10, 2020.

[5] H. Weber, C. Glasmacher, M. Schuldes, N. Wagener, and L. Eckstein, "Holistic driving scenario concept for urban traffic," *2023 IEEE IV*, pp. 1–8, 2023.

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

[7] C. Glasmacher, M. Schuldes, P. Topalakatti, P. Hristov, H. Weber, and L. Eckstein, "Scenario-based model of the odd through scenario databases," *https://vvm-project.de/https://www.vvm-projekt.de/en/publications*, 2023.

[8] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Körtke *et al.*, "6-layer model for a structured description and categorization of urban traffic and environment," *IEEE Access*, vol. 9, pp. 59 131–59 147, 2021.

[9] Gesamtverband der Deutschen Versicherungswirtschaft e. V., "Unfalltypen-katalog - leitfaden zur bestimmung des unfalltyps," *Unfallforschung der Versicherer*, 2016.

[10] E. de Gelder, J. Manders, C. Grappiolo, J.-P. Paardekooper, O. O. den Camp, and B. D. Schutter, "Real-world scenario mining for the assessment of automated vehicles," in *2020 IEEE 23rd ITSC*. IEEE, sep 2020.

[11] U. Economic and S. Council, "Ece/trans/wp. 29/2020/81: Proposal for a new un regulation on uniform provisions concerning the approval of vehicles with regards to automated lane keeping system."

[12] N. Epple, T. Hankofer, and A. Riener, "Scenario classes in naturalistic driving: Autoencoder-based spatial and time-sequential clustering of surrounding object trajectories," in *2020 IEEE 23rd ITSC*, 2020, pp. 1–6.

[13] A. Tenbrock, A. König, T. Keutgens, and H. Weber, "The conscend dataset: Concrete scenarios from the highd dataset according to alks regulation unece r157 in openx," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 174–181.

[14] M. Scholtes, M. Schuldes, H. Weber, N. Wagener, M. Hoss, and L. Eckstein, "Omegaformat: A comprehensive format of traffic recordings for scenario extraction," *FAS-Workshop*, pp. 195–205, 2022.

[15] C. Glasmacher, H. Weber, M. Schuldes, N. Wagener, and L. Eckstein, "Generation of concrete parameters from logical urban driving scenarios based on hybrid graphs," in *VEHITS - Conference*, 2023.

[16] B. Schütt, J. Ransiek, T. Braun, and E. Sax, "1001 ways of scenario generation for testing of self-driving cars: A survey," 2023.

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

[18] H. Weber, L. Eckstein, A. Tenbrock, A. König, J. Bock, and A. Zlocki, *Evidenzorientierte Ableitung von Grundszenarien für die Bundesautobahn*. Bundesanstalt für Straßenwesen (BASt), Jan. 2023.

[19] A.-M. Jacobo, U. Nobuyuki, Y. Kunio, O. Koichiro, K. Eiichi, and T. Satoshi, "Development of a safety assurance process for autonomous vehicles in japan," in *Proceedings of ESV Conference*, 2019.

[20] S. Geronimi and E. Arnoux. Adscene - a path to a european scenarios database for ads and adas specification, validation, and homologation. [Online]. Available: https://www.vvm-projekt.de/fileadmin/user_upload/Mid-Term/Presentations/VVM_HZE_EmmanuelArnoux.pdf

[21] M. Brini, E. Arnoux, B. Foyer, G. Bresson, L. Durville, F. Gaillard, and M. Pajon, "MOOVE & MOSAR Projects: a scenario library for designing & validating ADS," in *Driving Simulation Conference (DSC)*, Antibes, France, Sep. 2020.

[22] Safetypool™ scenario database. [Online]. Available: https://www.safetypool.ai/database

[23] G. Moioli, "Introducing blender 3.0," in *Introduction to Blender 3.0: Learn Organic and Architectural Modeling, Lighting, Materials, Painting, Rendering, and Compositing with Blender*. Springer, 2022, pp. 1–63.

[24] T. Brade and C. Glasmacher. Towards a sufficient odd completeness. [Online]. Available: https://www.vvm-projekt.de/final-event

[25] C. Glasmacher, M. Schuldes, H. Weber, N. Wagener, and L. Eckstein, "Acquire driving scenarios efficiently: A framework for prospective assessment of cost-optimal scenario acquisition," *arXiv preprint arXiv:2307.11647*, 2023.

[26] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1929–1934.

[27] esmini, https://github.com/esmini/esmini.

[28] SUNRISE consortium, https://ccam-sunrise-project.eu/.