Fitness Functions for Testing Automated and Autonomous Driving Systems

PrePRint for the proceedings of the 38th International Conference on Computer Safety, Reliability and Security 2019
The final authenticated publication is available online at
https://doi.org/10.1007/978-3-030-26601-1_5

Florian Hauer[1], Alexander Pretschner[1], and Bernd Holzmüller[2]

[1] Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany
{florian.hauer,alexander.pretschner}@tum.de
[2] ITK Engineering GmbH, Im Speyerer Tal 6, 76761 Ruelzheim, Germany
bernd.holzmueller@itk-engineering.de

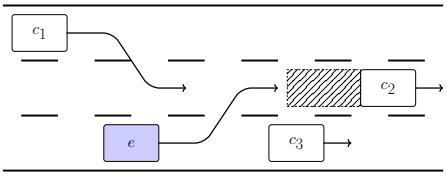
Abstract. Functional specifications and real drive data are typically used to derive parameterized scenarios for scenario-based testing of driving systems. The domains of the parameters span a huge space of possible test cases, from which "good" ones have to be selected. Heuristic search, guided by fitness functions, has been proposed as a suitable technique in the past. However, the *methodological challenge of creating suitable fitness functions* has not been addressed yet. We provide templates to formulate fitness functions for testing automated and autonomous driving systems. Those templates ensure correct positioning of scenario objects in space, yield a suitable ordering of maneuvers in time, and enable the search for scenarios in which the system leaves its safe operating envelope. We show how to compose them into fitness functions for heuristic search. Collision and close-to-collision scenarios from real drive data serve as a use case to show the applicability of the presented templates.

Keywords: System Verification · Automated & Autonomous Driving · Scenario-Based Testing · Search-Based Techniques

1 Introduction

Striving for highly automated and autonomous driving systems results in evermore complex and capable systems. Due to the complexity of these systems and the complexity and sheer number of possible scenarios, ensuring safety and functional correctness is a crucial challenge [9]. Since verification and validation by real test drives alone are practically infeasible [17], the focus shifts to virtual test drives. For virtual testing, scenario-based closed-loop testing in the form of X-in-the-Loop settings is used [16]. Such scenarios describe dynamic traffic situations to test the behavior of the automated or autonomous driving system. A whole set of such scenarios is encoded by a *parameterized scenario*. We show such a parameterized scenario for a highway pilot in Fig. 1. The ego vehicle e

accelerates from standstill and approaches car c_3, which is driving at lower velocity than e. e then changes to the middle lane, while simultaneously c_1 changes also to the middle lane behind e. During this scenario, e must not violate the safety distances, e.g. the one to c_2 (shaded area in Fig. 1). Each other car c_i, $i \in \{1, 2, 3\}$ has a parameter for its longitudinal starting position s_{0,c_i}, a starting time t_{start,c_i} for accelerating from standstill, and a desired velocity v_i it tries to reach and hold throughout the scenarios. In addition, the lane change of c_1 is triggered at a specific time, described by parameter t_{lc,c_1}. The domains of these ten parameters span a ten-dimensional space of possible test scenarios.



Parameter	Lower Bound	Upper Bound
Starting positions s_{0,c_i} [m]	0	300
Starting times t_{start,c_i} [s]	0	3
Target velocities v_i [km/h]	80	140
Lane change time t_{lc,c_1} [s]	6	13

Fig. 1. Parameterized highway scenario with ten parameters and their domains

Most scenarios in this space are not useful test cases, however. In some scenarios, e will not even perform a lane change; will perform it in front of c_2 instead of behind it; or c_1 performs its lane change several seconds later than e. Instead, "good" test cases need to be identified within the parameter space. In one interpretation of "good" test scenarios, a correct system *approaches* safe operating limits, and a faulty system violates them. Existing works suggest the use of search-based techniques. These were successfully applied for testing classic advanced driver assistance systems (SAE levels 1&2 [14]), e.g. a parking assistant, an adaptive cruise control, an emergency braking system, and their combination.

Those works focus on technical aspects, e.g. on how to improve the search algorithm, and assume the fitness functions to be given or created ad-hoc. This was an important, and successful, first step. Because these search-based techniques are so promising, we want to apply them to testing automated and autonomous driving systems of SAE levels 4&5 [14]. Such systems are fundamentally different, as they take over the complete driving task including decision making and executing active maneuvers in dynamic traffic scenarios. Thus, the variety of different possible parameterized scenarios is huge, which requires the definition of many different fitness functions. However, formulating fitness functions correctly is difficult, time-consuming, and requires experience. Wrongly derived fitness functions leave "good" test cases unidentified, which might even lead to wrong conclusions about the test results. It seems clear that creating fitness functions ad-hoc, as done in the past, is not sufficient. For the derivation of fitness functions at large scale, methodological guidance for test engineers is needed.

The **contribution of this paper** is the following: We provide such guidance in the form of a set of fitness function *templates* for testing automated and autonomous driving systems in dynamic traffic scenarios with heuristic search. It is further explained how those templates can be easily combined and applied to identify "good" test cases for complex scenario types.

§2 explains scenario-based testing and the application of search-based techniques in this domain. The templates are described in §3, before §4 presents ways to combine them. An application is provided in §5. We discuss related work in §6 and conclude in §7.

2 Scenario-Based Testing with Search-Based Techniques

In scenario-based testing of automated and autonomous driving systems, the goal is to test the behavior of such systems in dynamic traffic situations. A multitude of different scenario types exist. Several sources of information are used for the identification of those types, e.g. requirements, safety analysis, functional specifications, traffic rules, and real (test) drives. For each scenario type, one or more parameterized scenarios are derived, each describing a set of test cases. Generalizing and adapting the formalism of [2] and [10], we define a parameterized scenario as (X, V, D), where X is the data set that describes the scenario type (e.g. lane change) and context (e.g. two-lane highway). It can be described using the OpenScenario [1] or CommonRoad [5] formats. The variables $v_i \in V$ $(i \leq n)$ are parameters (e.g. velocities of traffic participants) with their domains $D_i \in D$. Assigning a value to each v_i yields a single test case. The domains in D span an infinite search space $A = D_1 \times D_2 \times \ldots \times D_n \subset \mathbb{R}^n$ of test cases.

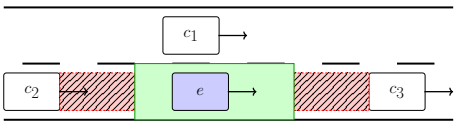


Fig. 2. Example of a simple safe operating envelope (green plain rectangle) bounded by the necessary safety distances (red shaded rectangles) and lane markings

The simulated scenario describes input and environment conditions of a **test case**. The expected behavior of continuous systems is described with the help of domains and thresholds. In this context, a safe operating envelope is used (Fig. 2). Inside the envelope, the system is allowed to freely optimize its performance [9], and as long as it does not leave the envelope, it is considered safe. By that the safe operating envelope provides a description of safe system behavior. It depends on the scenario and changes over time during the scenario. Recent works, e.g. the responsibility-sensitive safety (RSS) model [15], the safety force field model [11] as well as other formal models [13] presented such envelopes. These works provide a model of safe system behavior even for scenarios, in which the system alone cannot guarantee complete safety, as other traffic participants may still cause accidents. In the spirit of limit testing, we define a **"good" test case** as follows (see [12]):

> *A "good" test case can reveal* potentially *faulty system behavior. That means in a "good" test scenario, a correct system approaches the limits of the safe operating envelope, and a faulty system violates them.*

A **fitness function** $f : A \to W$ assigns every test case a **quality** value $w \in W$, which depends on the observed behavior of the system under test in the respective test case. It is important that a total order on the fitness values is preserved, such that a scenario gets a better quality value than another if it is a better test case. If the search space A and the quality function f are created accordingly, then search-based techniques may be used to find the "good" test cases in the following way (see Fig. 3):

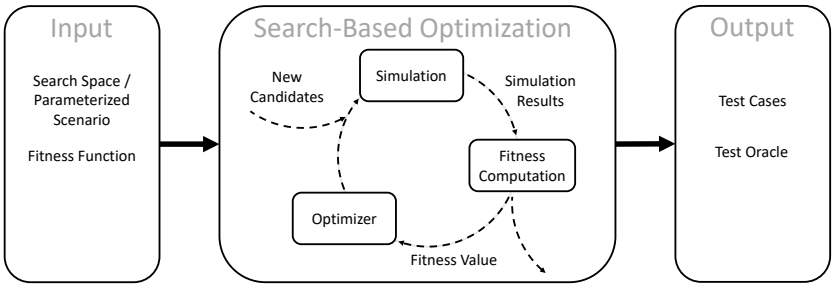


Fig. 3. Search-based techniques for scenario optimization

An initial set of scenario candidates is created either by reusing existing scenarios, by using manually created ones by experts, or by generating them randomly. These candidates are then executed in a simulation and the simulation results are evaluated by the fitness function, which returns a quantitative quality measure for the respective scenario. According to these fitness values, the optimization algorithm tries to adapt the parameter values in order to obtain scenarios of better quality. This iteration may be continued until a maximum number of iterations is reached, the assigned computation time is spent or the optimizer fails to find a better solution. This means that during the optimization process, the system is usually tested in one test case per fitness function evaluation, depending on the applied optimization technique. In the ideal case, search-based techniques would find the global optimum, which is the best scenario. This scenario is called worst-case. In the case that the system does not leave the safe operating envelope in the worst-case, it is considered to be safe.

In the following, templates are presented, which may be combined to fitness functions. For this work, the search space is assumed to be given, e.g. we use a parameterized scenario created by a domain expert.

3 Fitness Function Templates

In order to capture all potential scenarios, we present templates to aim for qualitative and quantitative test goals. Our goal is to find test cases, in which the system violates the safe operating envelope, e.g. by coming below a distance threshold. We call this a quantitative test goal, since a quantitative value (e.g. a distance between cars) is used to assign a fitness value. We present a suitable template to search for a violation of the safe operating envelope.

However, search spaces usually contain many scenarios in which a desired system behavior, e.g. a lane change, does not take place because the necessary context to provoke it does not occur. For instance, there is no lane change if there is no car to be overtaken. In theory it might be possible to only use a quantitative test goal and search for the violation of a safe operating envelope in a search space covering all possible scenarios. However, in practice this is undesired for several reasons. Scenario types (e.g. lane change, cut-in) are human-interpretable; testing every type on its own provides information about the quality of the system behavior in those specific scenarios. Further, testing these interpretable scenario types will be required by certification authorities. Lastly, such a theoretical search space that contains all possible scenarios is high-dimensional and complex. The search for a safety violation would be difficult - or even practically infeasible - for current search-based techniques. Thus, we need to ensure that the scenario description encodes the relevant parts of the context. Those are called qualitative test goals, since the mere existence of the relevant circumstances is used to assign a fitness value.

For dynamic scenarios, two aspects are of fundamental importance: space and time. Scenario objects need to be at the correct location at a specific moment, e.g. one car should be ahead of another. Furthermore, scenario events need to take place at the right moment in time, e.g. two cars should change the lane simultaneously. Since the (dynamic) behavior of the ego vehicle is unknown a-priori, the correct timing of maneuvers and positioning of scenario objects cannot be established statically and a-priori, e.g. by setting suitable parameter domain boundaries. However, incorporating such desired qualitative test goals into fitness functions is possible. We hence present specific templates for timing and positioning to ensure that such qualitative goals are fulfilled. During optimization those templates identify the scenarios that fulfill the qualitative test goals. Among them the best scenario is searched with the template that aims at the quantitative test goal. Note that in this work, **minimization** is used for optimization purposes. In the following, we will explain the generic idea first, before transferring it to templates for automated and autonomous driving systems.

3.1 Template for Testing Against Safe Operating Envelopes

We start with a very basic, simple, and intuitive template. Even though most of the existing works in this domain do not state it explicitly, their idea is to measure a certain system behavior and identify a test case in which this system behavior exceeds a threshold. For the case of a constant threshold, a qualitative generic example is provided in Fig. 4.

The blue time series describes a system behavior and the red line a threshold that must not be violated. This means the maximum value of the blue curve must not be greater than the threshold. During an optimization process, it is desired that better and better scenarios are found, which means that the maximum of the blue curve gets closer and closer to the red line or even surpasses it. The following fitness function idea may be used to achieve the described search behavior (assuming minimization): $f_{\text{idea},1} = -\max(\textit{blue curve})$

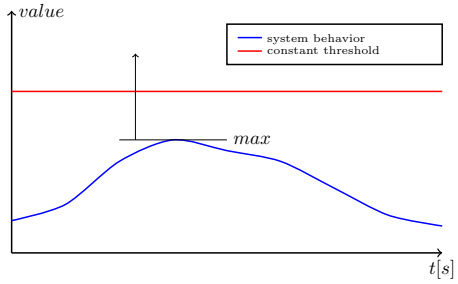


Fig. 4. Generic case of testing system behavior against a constant threshold

Now, this idea is transferred for testing automated and autonomous driving systems in dynamic traffic scenarios. As described in Section 2, instead of a constant threshold, a safe operating envelope is used, e.g. as presented in recent works [15,11,13]. Those works express safety often as a safety distance in time or space, which is usually depending on velocities of and relative positions among cars and, thus, is changing over time. We use *safeDist* as a placeholder for the computation of a safety distance according to such a safe operating envelope. We stick to the example of Fig. 1, but for the sake of simplicity, only c_2 and the safety distance to it are considered on a single lane for now (see Fig. 5):

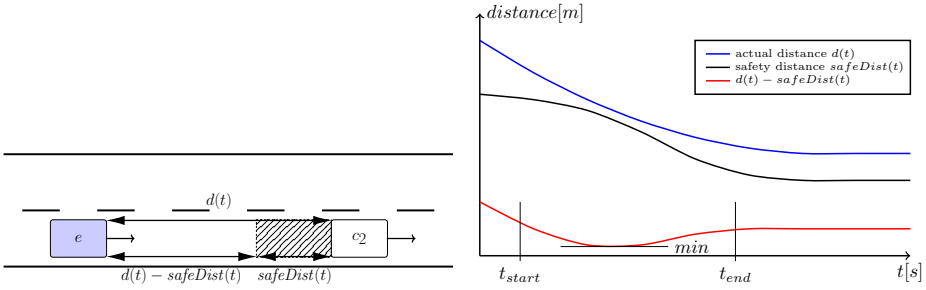


Fig. 5. Schematic depiction of a safety distance that should not be violated

The ego vehicle e is approaching another vehicle c_2, which is driving at lower velocity. Once e gets closer, it will reduce its velocity until it reaches the velocity of c_2. During this period, e must not violate the safety distance. Applying the classic idea $f_{\text{idea},1}$ as fitness function would mean that the scenario is searched, in which the distance d between e and c_2 gets smallest. However, a small d does not necessarily mean that the *safeDist* threshold is violated, since safety distances might be even smaller (relatively speaking) in scenarios with low velocities. One cannot conclude by the achieved fitness value whether the safe operating envelope has been violated or not. The dynamically changing safety distance has to be included into the template:

Template 1: $$f_{\text{template},1} = \min(d(t) - \mathit{safeDist}(t)) \tag{1}$$

The difference of $d(t)$ and $\mathit{safeDist}(t)$ is denoted as the remaining buffer until violation of the threshold (see image 5). Within the scenario, the minimum

remaining buffer is used as characteristic value, since it is the most dangerous moment. By applying this template, search techniques will identify the scenario in which the minimum of the remaining buffer is smaller than the minimum remaining buffer in all other scenarios. This has the side effect that the following test oracle can be applied for this template: If the remaining buffer is greater or equal than 0 even in the worst-case scenario, the system did never enter and, thus, never violate the safety distance. It even kept an additional distance equal to the remaining buffer in the worst-case. It never left the safe operating envelope and it is considered safe. If the remaining buffer is negative, a faulty system behavior is revealed. In this case, the absolute value is the amount by which the system violated the safety distance. Using this template, an argumentation basis for the release process is provided by making the system behavior measurable. With the help of this measurement, systems can be compared with respect to their performance in the system-specific worst-case.

3.2 Templates for Ensuring Qualitative Test Goals

General Idea to Ensure Qualitative Test Goals. A specific scenario does or does not satisfy the desired qualitative test goals. The following templates can be used to ensure such goals. By combination of multiple templates (§4), multiple qualitative test goals can be fulfilled. In the case of non-fulfillment of a goal, we assign the value m as fitness value, which has to be greater than any value that corresponds to a qualitative test goal being fulfilled. If m is constant, the optimizer will perform like random selection. However, we want to apply search-based techniques to identify scenarios that satisfy the desired qualitative test goals. Thus, this m should be a gradual measurement to provide a ranking among the scenarios that do not fulfill this qualitative test goal. In order to gradually reach a "fulfilling" scenario, a measurement is used for how far a scenario is away from fulfilling the goal. Since the mere fulfillment of the qualitative test goal is sufficient, every scenario that does so is equally good and, thus, receives the same constant fitness value, e.g. 0:

$$f_{\text{idea},2} = \begin{cases} m, & \text{qualitative test goal not fulfilled} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Assume a time series (blue curve in Fig. 6), which serves as input for the computation of m. At a specific time t_1, a qualitative test goal should be fulfilled. Fulfillment means that the value of the time series $value(t_1)$ at t_1 is in between the red thresholds z_{min} and z_{max}. Note that in general those thresholds do not need to be constant.

If $value(t_1)$ is outside the area described by the thresholds, m is the distance of $value(t_1)$ to the closer threshold to reach the area in between. During the optimization, $value(t_1)$ would approach the area. To avoid having one fitness

function per threshold, the mean of the thresholds is chosen:

$$f_{idea,3} = \begin{cases} \left| \frac{z_{min}+z_{max}}{2} - value(t_1) \right|, & value(t_1) \text{ outside} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

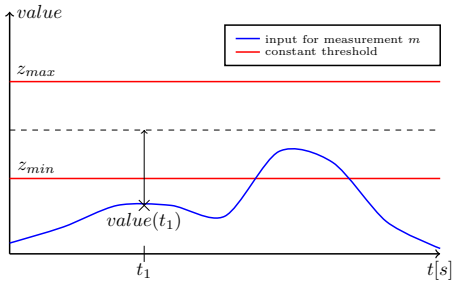


Fig. 6. Depiction of the general idea: The value of a curve at a specific moment has to be within a specific domain.

Template for Correct Positioning of Scenario Objects. This general idea is now transferred to a template. It ensures that scenario objects, e.g. cars, are correctly located relative to each other at a specific moment in time during the scenario. In Fig. 7, a scenario is depicted in which the ego vehicle e and the other cars c_1, c_2 are driving on two lanes next to each other. Assume that the qualitative test goal is that e is located in between c_1 and c_2 at a specific moment t_{event}. This might be desired in the case that e should perform a lane change into the gap bounded by c_1 and c_2.

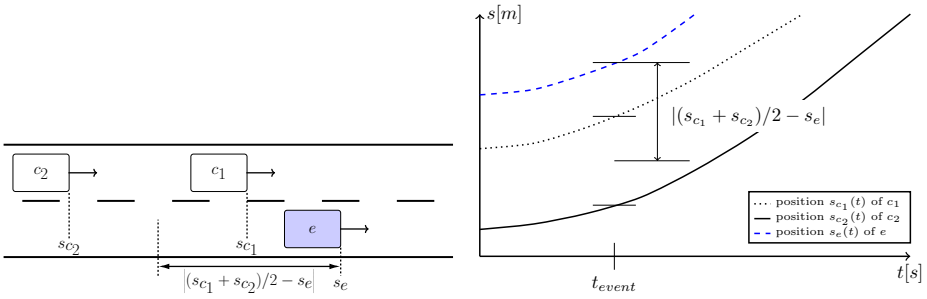


Fig. 7. Qualitative test goal: e should be located in the gap at t_{event}

The position of the ego vehicle s_e is used to compute the measurement m. The positions of the other cars s_{c_1}, s_{c_2} serve as thresholds. Note that in contrast to above, here the thresholds are not constant. The transferred template looks

as follows:

Template 2a: (4)

$$f_{template,2a} = \begin{cases} \left| \frac{s_{c_1}(t_{event}) + s_{c_2}(t_{event})}{2} - s_e(t_{event}) \right|, & e \text{ not in between } c_1 \text{ and } c_2 \\ 0, & \text{otherwise} \end{cases}$$

During the optimization, the structure of the template will bring e closer and closer to the gap until it is in the gap. However, as it is the case for the introductory example in Fig. 1, there might not be a gap. Only a single other car is of interest for relative positioning. The ego vehicle should be located behind c_2 for its lane change. This is reduced to the situation of Fig. 8.

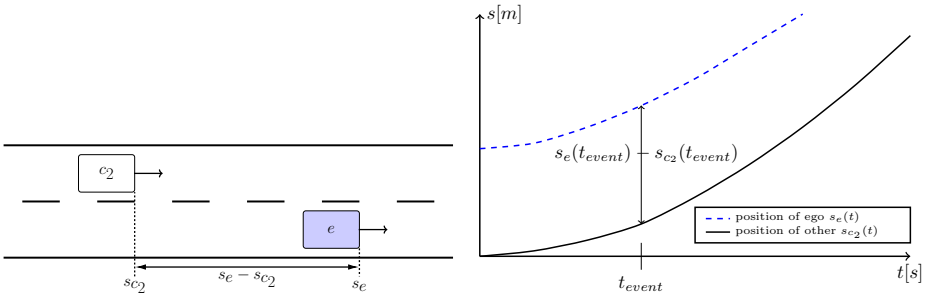


Fig. 8. Qualitative test goal: e should be located behind c_1 at t_{event}

In the case that e is ahead of c_2, the distance between them is used as measurement m. Since there is only one threshold ("behind of"), there is a slight difference to template 2a. This simplifies the template to the following, where only the distance to the one threshold is used:

Template 2b: (5)

$$f_{template,2b} = \begin{cases} s_{c_2}(t_{event}) - s_e(t_{event}), & s_e(t_{event}) < s_{c_2}(t_{event}) \\ 0, & \text{otherwise} \end{cases}$$

Template for Correct Timing of Scenario Events. So far, a template for the search of safe operating violations as well as templates for correct positioning of scenario elements were discussed. In the following, a template for timing is presented. It can be used to ensure that events, e.g. the start of a maneuver, are happening at the right moments in time relatively to each other. In the example of Fig. 1, the ego vehicle and the c_1 are supposed to perform their lane changes onto the middle lane simultaneously. This means that c_1 starts its lane change during the lane change of e. This is resembled in Fig. 9.

To allow c_1 to start its lane change even a bit before e, an offset Δt_1 can be used. In general, also an offset Δt_2 is possible, even though here it is set to 0. A $\Delta t_2 > 0$ would mean that c_1 starts lane changing after e already completed its lane change. The general idea of $f_{idea,3}$ is adjusted to yield a template for

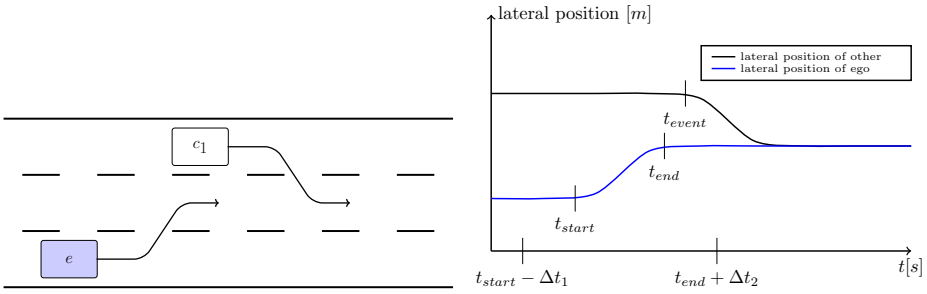


Fig. 9. Qualitative test goal: Lane changes should happen simultaneously

timings. However, this time the thresholds are not on the vertical axis as it is the case for the location templates, but on the horizontal one. The thresholds are the start t_{start} and the end t_{end} of the ego vehicle's lane change. In the case that the start of the other vehicle's lane change is not in between $t_{start} - \Delta t_1$ and $t_{end} + \Delta t_2$, the distance to the middle of the interval is chosen. The template for timing looks as follows:

Template 3: (6)

$$f_{template,3} = \begin{cases} \left| \frac{t_{start} - \Delta t_1 + t_{end} + \Delta t_2}{2} - t_{event} \right|, & t_{event} \text{ not in between bounds} \\ 0, & \text{otherwise} \end{cases}$$

4 Combining Templates

We have presented several templates, each addressing a specific aspect. In the following, it is described how those templates can be combined to a fitness function that can be used by search-based techniques to yield complex scenarios. There are two possibilities: Combining the set of templates to a single fitness function allows the usage of single-objective optimizers, while for multi-objective search, the fitness functions stay separated.

4.1 Combination for Single-Objective Search

The templates are nested into each other with the help of case distinctions. The innermost level in the nesting is a template that measures the behavior of the system with respect to a safe operating envelope; it aims for a quantitative test goal. The outer levels of nesting are templates for qualitative test goals (e.g. positioning and timing), which need to be fulfilled for the inner ones. Each level consists of one template returning the measurement m as described above. Instead of returning 0 in the case that the qualitative test goal is fulfilled, the measurement m of the next inner level is returned. This structure causes the optimizer to approach the search in steps. First, scenarios are searched that are of the desired form. Among those, the best scenario is identified for testing against a safe operating envelope. To ensure the necessary total order of fitness values, offsets are added to all levels of nesting except for the most inner one.

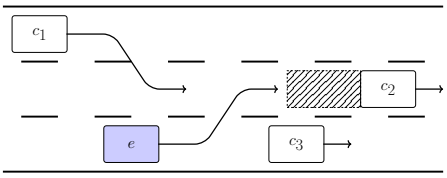
This offset needs to be greater than the maximum value of the next inner level. A simple overapproximation of the sum of the m of the next inner level plus the offset of the next inner level is sufficient.

4.2 Combination for Multi-Objective Search

Most likely, there are some goals that are not dependent on each other, meaning that each of those independent goals can be fulfilled without the constraint that the others need to be fulfilled. For instance in the introductory example in Fig. 1, the goal that "e performs its lane change behind c_2" can be fulfilled even though the goal that "c_1 performs its lane change simultaneously with e" is not fulfilled. In contrast to the usage of single-objective search, independent goals can be optimized simultaneously with multi-objective search. Multi-objective search optimizes a vector x of fitness values x_i instead of a single fitness value. The concept of Pareto optimization is used. A vector x is better than another vector y if all $x_i \leq y_i$ and at least one $x_i < y_i$. Each x_i is computed by a single template f_j, which may depend on one or more $f_k, j \neq k$. The f_j that are dependent on other $f_k, j \neq k$ need to be adjusted in the following way: In the case that at least one of the f_k is not 0, which means that the qualitative test goal connected to at least one of the f_k is not fulfilled, x_j is set to a very bad, high value. Step by step, the preliminary qualitative test goals will get fulfilled before the remaining test goals are optimized.

5 Application of the Templates

Since many car manufacturers and suppliers are currently developing a highway pilot system or a comparable system, such a system is chosen for demonstration purposes. It has to cope with all possible situations on the highway and does not require the driver to take over in critical situations. Therefore, the highway pilot is considered to be an automated driving system of SAE's level 4 [14]. Many natural driving studies have been conducted to gather data for further understanding of road traffic and the driver's task (e.g. [8]). The database of the biggest one [8] got analyzed for near-collision and collision recordings on highways. The findings were grouped to 24 scenario types [18]. We used those as use cases for the presented templates. In fact, the example of Fig. 1 and Fig. 10 is one of those scenario types. The presented templates ensure that maneuvers happen at the right moment and objects are located correctly, while another template searches for violations of the safe operating envelope. Using these templates, we were able to create suitable fitness functions for all of those scenario types. Since those are the near-collision and collision scenarios, they are the most critical ones. By at least covering those 24 scenarios with the templates, we argue that the presented set of templates is sufficient for most of the critical highway traffic scenarios. The following depicts the ease of use of the templates by applying them to the most complex scenario of the 24, which is the introductory example of Fig. 1 and Fig. 10. A variety of other scenarios is contained in this one, e.g. a lane change of the ego vehicle behind another car without further surrounding cars.



Parameter	Lower Bound	Upper Bound
Starting positions s_{0,c_i} [m]	0	300
Starting times t_{start,c_i} [s]	0	3
Target velocities v_i [km/h]	80	140
Lane change time t_{lc,c_1} [s]	6	13

Fig. 10. Most complex (close-to-)collision scenario of the reduced set of scenarios from the database analysis [18]

For this scenario, several fitness functions are needed:

- The lane change of e needs to happen, for which a constant template is used. For the given search space, a constant measurement m is not problematic, since many candidates contain a lane change of e.

$$\alpha = \begin{cases} \infty, & e \text{ does not change lanes} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

- The lane change of e needs to happen **behind** c_2, which indicates the use of the positioning template. Let the moment, when e gets past the lane markings between the starting lane and the target lane, be denoted as $t_{e,start}$.

$$\beta = \begin{cases} s_e(t_{e,start}) - s_{c_2}(t_{e,start}), & s_{c_2}(t_{e,start}) < s_e(t_{e,start}) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

- The lane change of c_1 needs to happen **behind** e. Again, the template for positioning is used. Let the moment, when c_1 gets past the lane markings between the starting lane and the target lane, be denoted as $t_{c_1,start}$.

$$\gamma = \begin{cases} s_{c_1}(t_{c_1,start}) - s_e(t_{c_1,start}), & s_e(t_{c_1,start}) < s_{c_1}(t_{c_1,start}) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

- Lane changes of e and c_1 need to be **simultaneously**, for which the timing template is used. Let the moment, when e and c_1 are fully on the target lane, be denoted as $t_{e,end}$ and $t_{c_1,end}$. If either $t_{c_1,start} + \Delta t_1 < t_{e,start}$ or $t_{e,end} < t_{c_1,end} - \Delta t_2$ is true, the lane changes are not considered to be simultaneous anymore. Δt_1 is set to 1s to allow for an earlier start of the lane change of c_1, while Δt_2 is set to 0s such that c_1 does not finish the lane change before e starts changing lanes.

$$\delta = \begin{cases} \left| \frac{t_{e,start}-1+t_{e,end}+0}{2} - \frac{t_{c_1,start}+t_{c_1,end}}{2} \right|, & \text{not simultaneous} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

- We need to search for a **violation of the safety distance**.

$$\epsilon = \min(s_{c_2}(t) - s_e(t) - \mathit{safeDist}(t)) \qquad [t_{e,start}, t_{e,end}] \tag{11}$$

The combined fitness function for single-objective search does look as follows. Powers of ten are used as offsets o_i, e.g. $o_1 = 10^3$ and $o_2 = 10^4$.

$$f_{single} = \begin{cases} \alpha + o_4, & e \text{ does not change lanes} \\ \begin{cases} \beta + o_3, & s_{c_2}(t_{e,start}) < s_e(t_{e,start}) \\ \begin{cases} \gamma + o_2, & s_e(t_{c_1,start}) < s_{c_1}(t_{c_1,start}) \\ \begin{cases} \delta + o_1, & \text{not simultaneous} \\ \epsilon, & \text{otherwise} \end{cases} \end{cases} \end{cases} \end{cases} \tag{12}$$

For an application of multi-objective search, the templates need to be changed, e.g. ϵ can only be computed if all qualitative test goals underlying the other templates are fulfilled.

$$\widetilde{\epsilon} = \begin{cases} \infty, & \alpha + \beta + \gamma + \delta > 0 \\ \epsilon \end{cases} \tag{13}$$

Incorporating the dependencies also in the other templates yields the final vector of fitness values. β, γ, and δ are independent of each other; they only depend on α. In contrast to a combination for single-objective search as above, they can be optimized simultaneously when combined for multi-objective search. α stays unchanged as it does not depend on other templates.

$$f_{multi} = [\alpha \quad \widetilde{\beta} \quad \widetilde{\gamma} \quad \widetilde{\delta} \quad \widetilde{\epsilon}] \tag{14}$$

The actual technical application of search-based techniques is not the focus of this work as is has been done by various existing works. However, for interested readers, we provide supplementary material online at https://mediatum.ub.tum.de/1474281. Contained are two experiments that use the presented parameterized scenario and combined fitness function as well as videos of the worst-case scenarios identified by single- and multi-objective search during the experiments.

6 Related Work

Search-based techniques have been proposed for test scenario selection. The initial research presented the idea of applying search-based techniques for the functional testing of advanced driver assistance systems by testing a parking assistant [6] and a braking assistant [7]. Their setup is close to what we describe as scenario-based testing. Recently, machine learning was introduced to improve the performance of test case generation in this domain. For instance, with learning surrogate models the optimization speed may be improved [3] and with building decision-trees the test engineer receives information about the search space during test case selection [2]. Both works apply the presented techniques on an emergency braking system. For testing a feature interaction of an adaptive cruise control and an emergency braking system, search-based techniques are improved in a way that they search for multiple faulty interactions simultaneously [4].

While all these technical improvements are important and show great results, these works assume the fitness function to be given or create them ad-hoc, e.g. to test the interaction of some specific features [4]. This is, because those works focus on the technical aspect of the search-based techniques. Neither of them addresses the methodological aspect of how fitness functions are correctly created to allow for statements about safety, e.g. by testing against a safe operating envelope as for instance provided by recent works [15,11,13]. Additionally, the evaluation systems are rather reactive driver assistance systems of SAE level 1&2 [14] or combination of such. The provided fitness functions for those systems are mostly not applicable to higher automated systems (e.g. level 4&5) with decision making and active functionality (e.g. lane changing or overtaking) in complex dynamic traffic scenarios, which require the fulfillment of qualitative test goals. This motivates the need for methodological guidance when deriving fitness functions.

7 Conclusion

We started by describing the necessity of suitable fitness functions to identify "good" test cases within huge search spaces, described by parameterized scenarios for automated and autonomous driving. A correct derivation of such suitable functions is crucial, but difficult. For the application of search-based techniques at larger scale for testing automated and autonomous driving systems, guidance for test engineers is necessary. In this work, we provide such guidance in form of templates and the means to combine them to fitness functions for complex traffic scenarios. To test against thresholds of a safe operating envelope, we presented a specific template which provides the test engineer with an automated oracle. Additional templates for relative positioning in time and space ensure that the optimizer identifies scenarios that fulfill the qualitative test goals. For combining the templates, we presented both a single and a multi-objective approach which make use of case distinctions to provide a total ordering on scenario candidates such that better scenarios are assigned to better fitness values. As an evaluation, we presented the application of the templates on the most complex (close-to-)collision highway scenario contained in the biggest natural driving study database (identified by [18]). We conclude that the presented templates provide a structured way for test engineers to formulate fitness functions to identify "good" test cases. Thus, this work adds a much needed methodological angle to the otherwise technical solutions.

The application of search-based techniques requires both a fitness function and a search space. The derivation of the search space is not discussed in this work. Similarly to the fitness function derivation, methodological guidance for the derivation of search spaces (parameterized scenarios) is of high interest. Both are difficult the creation of a suitable skeleton of a parameterized scenario and the identification of suitable parameters and their domains. Further, in addition to the methodological guidance presented in this work, an automated fitness function derivation would be very useful to support test engineers. Using

a suitable scenario description as input, the described combination for single- and multi-objective techniques might be automated.

References

1. OpenScenario 0.9.1 (2017). Tech. rep., OpenScenario Initiative - online at http://www.openscenario.org, retrieved 11th January 2019
2. Abdessalem, R.B., Nejati, S., Briand, L., Stifter, T.: Testing vision-based control systems using learnable evolutionary algorithms. In: Proceedings of the 40th International Conference on Software Engineering (ICSE 2018). ACM (2018)
3. Abdessalem, R.B., Nejati, S., Briand, L.C., Stifter, T.: Testing advanced driver assistance systems using multi-objective search and neural networks. In: 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 63–74 (2016)
4. Abdessalem, R.B., Panichella, A., Nejati, S., Briand, L.C., Stifter, T.: Testing autonomous cars for feature interaction failures using many-objective search. In: 33rd ACM/IEEE International Conference on Automated Software Engineering. pp. 143–154 (2018)
5. Althoff, M., Koschi, M., Manzinger, S.: Commonroad: Composable benchmarks for motion planning on roads. In: Intelligent Vehicles Symposium (IV), 2017 IEEE. pp. 719–726. IEEE (2017)
6. Bühler, O., Wegener, J.: Evolutionary functional testing of an automated parking system. In: Proceedings of the International Conference on Computer, Communication and Control Technologies (CCCT) and the 9th. International Conference on Information Systems Analysis and Synthesis (ISAS) (2003)
7. Bühler, O., Wegener, J.: Evolutionary functional testing. Computers & Operations Research **35**(10), 3144–3160 (2008)
8. Hankey, J.M., Perez, M.A., McClafferty, J.A.: Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets. Tech. rep., Virginia Tech Transportation Institute (2016)
9. Koopman, P., Wagner, M.: Challenges in autonomous vehicle testing and validation. SAE International Journal of Transportation Safety **4**(1), 15–24 (2016)
10. Mullins, G.E., Stankiewicz, P.G., Gupta, S.K.: Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. In: IEE International Conference on Robotics and Automation (ICRA). pp. 1443–1450 (2017)
11. Nister, D., Lee, H.L., Ng, J., Wang, Y.: The safety force field. online at https://www.nvidia.com/content/dam/en-zz/Solutions/self-driving-cars/safety-force-field/the-safety-force-field.pdf, retrieved 10th May 2019
12. Pretschner, A.: Defect-based testing. In: Dependable Software Systems Engineering (2015)
13. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., et al.: Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In: International Conference on Integrated Formal Methods. pp. 50–66. Springer (2017)
14. SAE: Definitions for terms related to on-road motor vehicle automated driving systems. J3016, SAE International Standard (2014)
15. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. arXiv:1708.06374 (retrieved 5th May 2019)

16. Ulbrich, S., Schuldt, F., Homeier, K., Steinhoff, M., Menzel, T., Krause, J., Maurer, M.: Testing and validating tactical lane change behavior planning for automated driving. In: Automated Driving, pp. 451–471. Springer (2017)
17. Wachenfeld, W., Winner, H.: The release of autonomous vehicles. In: Autonomous Driving, pp. 425–449. Springer (2016)
18. Zhou, J., del Re, L.: Reduced complexity safety testing for adas & adf. IFAC-PapersOnLine **50**(1), 5985–5990 (2017)