Test Coverage of Automated Robotic Systems in Open World Environments

Lukas Westhofen[1(✉)], Till Schallau[2], Dominik Schmid[2], Stefan Naujokat[2], Falk Howar[2,3], and Daniel Neider[2,4]

[1] German Aerospace Center, Institute of Systems Engineering for Future Mobility, Oldenburg, Germany
lukas.westhofen@dlr.de
[2] TU Dortmund University, Dortmund, Germany
[3] Fraunhofer Institute for Software and Systems Engineering, Dortmund, Germany
[4] Center for Trustworthy Data Science and Security, Dortmund, Germany

Abstract. Automated robotic systems operating in real-world environments require thorough test campaigns before deployment, in which engineers assess whether the system is actually exposed to critical scenarios. A well-established way to judge the efficacy of test campaigns in controlled environments is scenario coverage: It quantifies the quality of recorded data from test campaigns w.r.t. a set of (critical) scenario classes of interest. Challenges arise when transferring coverage from controlled to open environments, as it may no longer be exactly determined to which scenario class the recorded test data belongs to: sensors offer limited observability, e.g., by occlusions or hardware failures, and specifications might be inherently vague, e.g., via imprecise traffic regulations. We leverage the Open World Assumption to formally extend scenario coverage to such ambiguous test data and present algorithms for computing guaranteed lower and upper bounds on coverage. Whereas deciding the lower bound problem is D^p-complete, the upper bound can be computed in polynomial time. We extend an existing coverage software tool with two approaches for incorporating the Open World Assumption, grounded in LTL$_f$. Our evaluation shows that meaningful statements on scenario coverage are feasible, even under intricacies of the real world.

Keywords: Testing · Robotics · Safety · Automated Driving · Scenario Coverage · Temporal Logics

1 Introduction

Verification of highly automated robotic system relies heavily on testing due to complex and uncontrolled operating environments. This forces manufacturers to test in the intended operating domain already during development. The collected data are then used to evaluate system performance. Prominent examples are the large-scale test campaigns by automated driving companies, such as Waymo.

When testing, a central question arises: Did the observed test runs actually expose the system to critical situations? An effective way to quantify this

A. Sampaio and M. Stoelinga (Eds.): FM 2026, LNCS 16556, pp. 130–150, 2026.
https://doi.org/10.1007/978-3-032-26204-2_7

question is through scenario coverage [15, Section 3.3], in which engineers specify a set of scenario classes to cover the space of relevant, e.g., safety-critical, environmental inputs to the system, as mandated by ISO 21448 [20]. In Germany, this is even legally required for automated driving (which will act as an example use case throughout this work): "test cases must provide sufficient test coverage for all scenarios, test parameters, and environmental influences. The coverage must be justified [...]" [11]. Coverage of scenario classes, i.e., particularly interesting type of test inputs, by a recorded data set then refers to the proportion of the scenario classes for which at least one instance was observed. The goal is to identify testing gaps, which can then be re-tested more exhaustively. Recent approaches have examined scenario coverage in environments with almost-perfect knowledge, such as bird-eye simulations or drone data [13,29].

In this paper, we push coverage from laboratory to real-world settings. This presents a challenge: both data and specifications might be incomplete. Under these assumptions, the scenario class of a recording cannot always be exactly determined: First, there is no perfect ground truth, i.e., the recorded data is incomplete. This can be caused by perception characteristics such as occlusions, sensor failures, and lossy post-processing. Second, even if all relevant information is available, there might be incomplete specifications of scenario classes due to complexity and vagueness. Causes include intentionally fuzzy definitions in legal texts (such as safety distances) [10,30] and partial knowledge of engineers.

To account for both incomplete data and knowledge, this work incorporates the Open World Assumption (OWA) into scenario coverage. Intuitively, the OWA states that we cannot infer the truth of a statement from the absence of knowledge about the inverse statement [22], e.g., we cannot state that no vehicle is present from not knowing about any vehicle. This leads to situations in which neither a statement nor its inverse hold, representing absence of knowledge about the actual truth. We can use this OWA to propagate both incomplete data – by not inferring from the absence of data – and incomplete knowledge – by allowing for "gray zones" in specifications – into coverage. Observed scenarios can then possibly belong to one of multiple scenario classes. For example, we might observe a record where it is unknown if a passed vehicle is parking. It must be either parking or not parking; however, based on the available data and specification, it can be impossible to determine exactly its state.

The main contribution of this paper is to extend the classic coverage problem (introduced in Sect. 2) by an OWA: Instead of unambiguous assignments to scenario classes, we now also allow for gray zones. Since coverage cannot be exactly determined anymore, we update the classic problem to now yield lower and upper approximations in Sect. 3. Naïve algorithms fail for both bounds. We hence analyze computational complexity and present efficient algorithms. In Sect. 4, we contribute two ways of incorporating the OWA in a temporal logic for scenario class specification. The first is based on standard propositional logic, while the second uses Description Logics (DLs). An implementation of both approaches allows an evaluation of OWA coverage on practically relevant configurations in Sect. 5. As we delineate in Sect. 6, this contribution is novel in the sense that,

to the best knowledge, all related works make the Closed World Assumption (CWA), i.e., assuming the available knowledge to be complete. While limitations of our OWA integration remain, we conclude in Sect. 7 that the approach is efficient and useful in real-world settings.

2 Preliminaries

We are faced with recorded results of test runs of the robotic system as a list of finite data traces $(d_1, \ldots, d_n)$. Each data trace d_i consists of data points that contain environment information, e.g., road infrastructure, as well as dynamic properties, e.g., positions and speeds of actors. Formally, $d_i = (\mathfrak{D}_1^i, \ldots, \mathfrak{D}_m^i)$, where $\mathfrak{D}_j^i \in \mathbb{D}$ is the recorded data at timestamp j for a (for now not further specified) set $\mathbb{D}$ of all possible data points (Sect. 4 gives two realizations of $\mathbb{D}$).

Based on the ISO 34504 terminology, our *scenario classes* correspond to *scenario categories* defined by sets of tags [21]. We model each tag as a Boolean predicate over a recorded test run, so class membership follows ISO's tag-inclusion semantics (conjunction of tag predicates). Coverage metrics therefore rely on an explicit abstraction that maps continuous, high-dimensional data traces to a finite tag vocabulary. This abstraction is a standard and necessary precondition of coverage analysis: it makes coverage computation and gap identification tractable, but it introduces the usual trade-off that overly abstract tags may hide relevant differences while overly fine-grained tags can reduce interpretability and scalability. Accordingly, any coverage statement is conditional on the chosen tags and the tag-assignment method. In this paper, we treat the tag vocabulary and classifiers as given engineering inputs and focus on making coverage robust when data or specifications are incomplete.

It is envisioned that the system is put under comparable challenging conditions in each instance of a scenario class – such as encountering a pedestrian at parking vehicles. To represent this formally, we use a finite set of tags T and a finite set of possible values V_t for each tag $t \in T$. The set of all values is $V = \bigcup_{t \in T} V_t$. This results in a list of *scenarios* $(s_1, \ldots, s_n)$ corresponding to the given test runs, where each s_i is a function from T to V.

As a running example, consider four tags $T = \{t_1, \ldots, t_4\}$. For simplicity, $V = \{0, 1\}$, where 0 encodes the absence and 1 presence of the tag in the data:

- $\mathbf{t_1}$ denotes whether a violation of the safety distance to a lead vehicle was observed.
- $\mathbf{t_2}$ holds if a pedestrian or bicycle is present.
- $\mathbf{t_3}$ addresses passing of parking vehicles (as opening doors or occluded pedestrians can be critical).
- $\mathbf{t_4}$ specifies a condition where cameras are susceptible to glare: The system drives towards a low sun for a considerable amount of time.

An example scenario $s_1 : t_1 \mapsto 0, t_2 \mapsto 0, t_3 \mapsto 0, t_4 \mapsto 1$ encodes no safety distance violation, present pedestrian or bicycle, or passing of parking vehicles, while the system drives towards a low sun.

For deriving s_i from a recorded data trace d_i, we assume a function from the set of all data traces of finite lengths to tag evaluations: $\Lambda_{T,V} : \bigcup_{h>0} \mathbb{D}^h \to V^T$ for tags T and values V, where $V^T = \{f : T \to V\}$. As we operate on data traces, we can apply temporal logics for classification. For now, we will not put further constraints on $\Lambda_{T,V}$ and assume that $\Lambda_{T,V}(d_i)$ was evaluated on all data traces. Section 4 gives possible implementations of $\Lambda_{T,V}$.

When running test campaigns, engineers are typically interested in how exhaustive the system has been tested after the data has been assigned to the predefined scenario classes. A possible way to assess the exhaustiveness is *scenario class coverage* (SCC). It has been examined under the Closed World Assumption (CWA) [29]. The CWA implies that the data are given as a perfect ground truth and no ambiguities exist in the specification. SCC is defined as the ratio of observed and all possible scenario classes for a list of scenarios $(s_1, \ldots, s_n)$:

$$\mathrm{SCC}(\mathbb{T}, V, s_1, \ldots, s_n) = \frac{|\{s_i \mid i \in \{1, \ldots, n\}\}|}{|\mathbb{T}|}, \tag{1}$$

where $\mathbb{T}$ is the space of all possible options. In the simplest case, $\mathbb{T} = V^T$ (which we assume in this work). Equation 1 allows quantifying the gap in which the system has not yet been tested and gives valuable feedback to engineers and authorities.

3 Coverage Under the Open World Assumption

The main contribution of our work is to extend scenario class coverage to the OWA, as the CWA is not always applicable, especially when testing robotic systems in real-world environments. The OWA can alleviate this problem by allowing the value of some tags to be unknown due to two reasons: The behavior of robotic systems is often hard to specify completely. Additionally, their perception is naturally limited due to sensor technology capabilities. As an example, consider the tag t_1 (safety distance violation). While several rules of thumb and approximating formulae exist, a gray area will always remain in the specification. Hence, t_1 must be considered under the OWA.

To formalize this, for each tag, we allow adding a unique symbol $*$ to V_t, where $s_i(t) = *$ indicates that $s_i(t)$ is not known and might take any value from the original tag values $V_t \setminus \{*\}$. In our example, we extend the values for the first tag, V_{t_1}, from $\{0, 1\}$ to $\{0, 1, *\}$ with $*$ encoding no observation of a safety distance violation. This can happen due to perception insufficiencies or fuzzy specifications. A similar reasoning can be applied to t_2 (presence of a pedestrian or bicycle) and t_3 (passing of parking vehicles). These tags will serve as our running example. For instance, in Sect. 5, we analyze their value distributions on a real-world dataset. Note that not all tags must be given the $*$ option, thus allowing to still partially rely on the CWA, e.g., if perfect information can be assumed, such as positioning and time for t_4. For tags with an "unknown" option, i.e., t where $* \in V_t$, on the other hand, $s_i(t) = *$ induces the set $V_t \setminus \{*\}$ as possible options, from which only one could have been true. We are thus

confronted with a set of possible options for a scenario s_i. This set contains all functions representing possible resolutions of $*$-entries of a tag t to some value in $V_t \setminus \{*\}$, and otherwise respecting the non-$*$-entries of s_i:

$$\mathsf{O}(s_i) = \left\{ o : T \to V \middle| \begin{array}{c} \forall t \in T.\ s_i(t) \neq * \to o(t) = s_i(t) \wedge \\ s_i(t) = * \to o(t) \in V_t \setminus \{*\} \end{array} \right\}.$$

In what follows, we operate directly on $\mathsf{O}(s_i)$ instead of s_i, which means that we use a list $S = (S_1, \ldots, S_n) = (\mathsf{O}(s_1), \ldots, \mathsf{O}(s_n))$ instead of $(s_1, \ldots, s_n)$. This translation creates, in the worst case, $|V|^{|T|}$ many entries for each observed scenario and is therefore exponential. Hence, all examined problems would directly be exponential as well. Assuming a constant tag size on the other hand allows us to uncover subtleties in the theoretical data complexity [34] of computing coverage under an OWA.

3.1 Definition of Lower and Upper Coverage Bounds

We can now update the definition of coverage under the CWA as given in Eq. 1 to our formalized OWA framework. This is not straightforward: due to the missing information, the numerator of the SCC cannot be determined exactly. However, it is still possible to identify correct lower and upper approximations to the unknown numerator of the SCC by appropriately selecting one of the possible options. For the lower bound, the goal is to select the options such that as many duplicates (i.e. identical scenario classifications) as achievable are created to calculate a pessimistic coverage guarantee. The upper bound on coverage, on the other hand, maximizes the diversity of the scenarios by avoiding duplicates to emulate the best possible bound, if possible. The idea of optimally selecting scenarios from the set of options is formally captured by the following two problems, representing the lower and upper bound, respectively.

Definition 1 (MinCover). *For a list of non-empty finite sets $S = (S_1, \ldots, S_n)$ over the universe $U = \bigcup_{i \in \{1,\ldots,n\}} S_i$, find k_{min} which is the smallest $k \leq n$ s.t. there is a function π with $|Im(\pi)| = k$ assigning each index i an element of S_i, i.e., $\pi : \{1, \ldots, n\} \to U$ where $\pi(i) \in S_i$ for all $i \in \{1, \ldots, n\}$.*

Definition 2 (MaxCover). *For a list of non-empty finite sets $S = (S_1, \ldots, S_n)$ over the universe $U = \bigcup_{i \in \{1,\ldots,n\}} S_i$, find k_{max} which is the largest $k \leq n$ s.t. there is a function π with $|Im(\pi)| = k$ assigning each index i an element of S_i, i.e., $\pi : \{1, \ldots, n\} \to U$ where $\pi(i) \in S_i$ for all $i \in \{1, \ldots, n\}$.*

Here, $\mathrm{Im}(\pi)$ refers to the image of π. Both problems require selecting exactly one element from each set S_i s.t. the number of unique, overall selected elements is minimal or maximal, respectively. The corresponding decision problems IsMinCover and IsMaxCover ask if $k \in \mathbb{N}$ is equivalent to k_{min} or k_{max}, respectively.

Consider our running example, where t_1 to t_3 are interpreted under the OWA. The following examples represent the i-th scenario as $(s_i(t_1), \ldots, s_i(t_4))$, e.g., $(1, 0, 0, 1)$, and, for ease of presentation, also refers to its list of tags with value

1, e.g., (t_1, t_4). Assume four recorded scenarios $s_1 = (0, 0, 0, 1)$, $s_2 = (1, 0, 0, 1)$, $s_3 = (*, 0, 0, 1)$, and $s_4 = (*, *, 0, 1)$. To determine both approximations, we derive all possibilities and compute coverage for all combinations (explicated values are marked bold):

$$\mathsf{O}(s_1) = \{s_1\} = \{(0,0,0,1)\} = \{(t_4)\} \quad \mathsf{O}(s_2) = \{s_2\} = \{(1,0,0,1)\} = \{(t_1,t_4)\}$$

$$\mathsf{O}(s_3) = \left\{ \begin{array}{l} (\mathbf{0},0,0,1) = (t_4), \\ (\mathbf{1},0,0,1) = (t_1,t_4) \end{array} \right\} \qquad \mathsf{O}(s_4) = \left\{ \begin{array}{l} (\mathbf{0},\mathbf{0},0,1) = (t_4), \\ (\mathbf{0},\mathbf{1},0,1) = (t_2,t_4), \\ (\mathbf{1},\mathbf{0},0,1) = (t_1,t_4), \\ (\mathbf{1},\mathbf{1},0,1) = (t_1,t_2,t_4) \end{array} \right\}$$

The worst case assigns s_3 and s_4 the same scenario as s_1 or s_2. The corresponding function π of Definition 1 can, e.g., assign $\pi(1) = \pi(3) = \pi(4) = s_1$ and $\pi(2) = s_2$, and hence $k_{min} = |Im(\pi)| = 2$. The best case assigns s_4 a unique scenario. However, s_3 must always be an existing scenario, as $s_1, s_2 \in \mathsf{O}(s_3)$. We can hence set $\pi(1) = \pi(3) = s_1$, $\pi(2) = s_2$, $\pi(4) = (t_1, t_2, t_4)$, and $k_{max} = |Im(\pi)| = 3$.

The values k_{min} and k_{max} can be used as a replacement of the numerator in Eq. 1. This allows us to give lower and upper approximations on the (indeterminable) SCC when given a list of observed scenarios $(S_1, \ldots, S_n)$ under the OWA:

$$\frac{k_{min}}{|\mathbb{T}|} \leq \mathrm{SCC}(\mathbb{T}, (s'_1, \ldots, s'_n)) \leq \frac{k_{max}}{|\mathbb{T}|},$$

where $s'_i \in S_i$ is the "real" ground truth of the i-th scenario that could not be determined. The computation of the denominator does not change under the OWA, i.e., we still require computing the size of the option space $|\mathbb{T}|$.

When testing robotic systems, the lower and upper bound can inform test engineers and approval authorities about the current test coverage state, even under incomplete specifications or imperfect perception. There are, however, practical differences between both bounds, and MinCover (k_{min}) might be argued as more relevant in practice than MaxCover (k_{max}). MinCover reports the minimum amount of scenarios possibly seen in the test run, and hence can be used for proving coverage against approval authorities. In Germany, such a coverage demonstration is required for a permit to operate an automated driving system, as argued in the introduction [11]. Contrarily, a low MaxCover can indicate a required increase in efforts to engineers during testing since it reports the best coverage possible when assuming maximum diversity in the observed scenarios. If it is known that the best case coverage is still below the designated target value, the tests will provably not suffice for a system release. While this is valuable input to guide the engineers' decisions, it is of a rather informative nature compared to the approval-enabling capabilities of the lower bound.

3.2 Computing the Lower and Upper Bound

We now analyze the complexity of computing both bounds. From a practical perspective, this especially allows understanding how the required computational

efforts depend on the data, which can grow extremely large as the test campaigns progress. As we will see, handling extensive test data is not a small feat.

A naïve algorithm is to compute coverage for all possible explicated combinations, which has a worst-case exponential complexity of $\mathcal{O}(|V|^{|T|\cdot n})$. Even if $|T|$ is constant, this is not a viable solution as n typically grows as the test campaign progresses, resulting in a complexity of $\mathcal{O}(2^n)$ already for binary tags. Preliminary experiments show that this brute-force algorithm is infeasible on small data: it did not terminate within one hour on 10^2 scenarios on standard consumer hardware for three OWA and four CWA tags (with equal probabilities for 0 and 1, and 10% probability for $*$-entries). Even a more sophisticated method involving highly efficient Satisfiability Modulo Theories (SMT) solving fails the same setup for 10^3 scenarios (more details are given online [41]).

The main finding of this work is that we can devise algorithms that perform theoretically *and* practically better than both brute force and SMT solving. We will see that IsMinCover is, in fact, not easily solvable: It is D^p-complete, where $D^p = \{L_1 \cap L_2 \mid L_1 \in \text{NP}, L_2 \in \text{coNP}\}$, i.e., the intersection of an NP- and coNP-problem. Contrarily, IsMaxCover can be computed in polynomial time. Hence, from a theoretical viewpoint, both the brute-force and SMT-based approach are suboptimal for computing the upper approximation. For the lower approximation, we show that – from a practical perspective – there is a problem in which the bound computation can be encoded more naturally than SMT, yielding practical improvements.

MinCover. IsMinCover is D^p-complete: First, note that it is the intersection of an NP-problem (checking whether there is a function with the mentioned constraints) and a coNP-problem (checking whether there is no such smaller function). Leveraging this duality, hardness follows by reduction from SAT-UNSAT, a classical D^p-complete problem [26]. A full proof is given in the online material [41].

Our computation of MinCover leverages analogies to finding minimal hitting sets, which is typically solved by unweighted partial MaxSAT. Let $X = \{x_1, \ldots, x_u\}$ be a set of variables and $\overline{X} = \{\neg x \mid x \in X\}$ their negations. A Conjunctive Normal Form (CNF) is a set of clauses $C = \{C_1, \ldots, C_v\}$ where each clause is a finite set of literals from $X \cup \overline{X}$. A solution to a CNF C is a function that satisfies all of its clauses, i.e., $\alpha : X \to \{0, 1\}$ where for all $C_j \in C$ there is some $l \in C_j$ s.t. $\alpha(l) = 1$ if $l \in X$ or $\alpha(x) = 0$ if $l \in \overline{X}$. In unweighted partial MaxSAT, we are given two CNFs C_h (hard clauses) and C_s (soft clauses, which have an implicit weight 1). The task is to find a solution to C_h which additionally satisfies a maximal number of clauses in C_s.

To solve MinCover via a MaxSAT solver, we set $C_h = S$. To ensure that the identified solution is minimal, soft clauses $C_s = \{\{\neg s\} \mid s \in U\}$ are added. This enforces solutions which maximize the number of excluded elements of U but still (due to the hard clause) hit each set at least once. The desired k_{min} is then the number of satisfied variables, i.e., $k_{min} = |\{s \in U \mid \alpha(s) = 1\}|$. Correctness is shown in the online material [41]. While MaxSAT is NP-hard

(more specifically, OptP-complete [23]), optimized algorithms exist and current approaches can offer a worst-case complexity $\mathcal{O}^*(1.2886^n)$ on our setting [42]. An exponential dependency on the data remains, yet this reflects a substantial improvement over $\mathcal{O}(2^n)$ required by the naïve brute-force approach. For our running example, we encode the four scenarios in S by the following CNFs:

$$C_h = \big\{\{(t_4)\}, \{(t_1, t_4)\}, \{(t_4), (t_1, t_4)\}, \{(t_4), (t_2, t_4), (t_1, t_4), (t_1, t_2, t_4)\}\big\},$$
$$C_s = \big\{\{\neg(t_4)\}, \{\neg(t_1, t_4)\}, \{\neg(t_2, t_4)\}, \{\neg(t_1, t_2, t_4)\}\big\},$$

The resulting maximal solution is indeed $\alpha((t_4)) = \alpha((t_1, t_4)) = 1$ and $\alpha((t_2, t_4)) = \alpha((t_1, t_2, t_4)) = 0$ (satisfying two of the four soft clauses), confirming $k_{min} = 2$.

MaxCover. Contrary to MinCover, MaxCover can be computed in polynomial time. For this, we reduce MaxCover to finding the maximum matching in bipartite graphs [19] (which we call MaxMatch). An undirected graph $G = (V, E)$ with $E \subseteq \{\{v_1, v_2\} \mid v_1, v_2 \in V\}$ is bipartite if there is a partition of V into V_1 and V_2 s.t. $E = \{\{v_1, v_2\} \mid v_1 \in V_1, v_2 \in V_2\}$. For a bipartite graph $G = (V, E)$, a maximum matching is a maximal set of edges $E_{max} \subseteq E$ s.t. for all $\{v_1, v_2\}, \{v_3, v_4\} \in E_{max}$ with $\{v_1, v_2\} \neq \{v_3, v_4\}$ it holds that $v_1 \neq v_3$ and $v_2 \neq v_4$. This problem can be solved in polynomial time, specifically, $\mathcal{O}(\sqrt{V}E)$ [19], and recently, even more optimized algorithms were proposed [7].

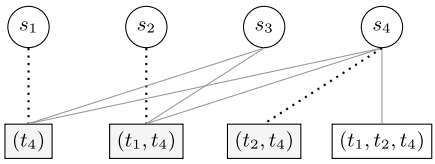


Fig. 1. G_S for the running example; the maximum cardinality matching is dotted.

To reduce MaxCover for $S = (S_1, \ldots S_n)$ over a universe U to MaxMatch, we define a graph $G_S = (\{s_1, \ldots, s_n\} \cup U, \{\{s_i, s\} \mid i \in \{1, \ldots, n\}, s \in S_i\})$ over the partitions $\{s_1, \ldots, s_n\}$ and U. Then, $k_{max} = |E_{max}|$, i.e., the cardinality of a maximum matching of G_S. The proof is available online [41]. In terms of our original inputs, this computation translates to a complexity of $\mathcal{O}(\sqrt{n + |U|} \sum_{S_i \in S} |S_i|)$, improving on the naïve and SMT-based approaches. This also implies the corresponding decision problem IsMaxCover to be in P. For the running example, the resulting graph is given in Fig. 1.

4 Realizing Open World Scenario Classification

In Sect. 3, we simply assumed unknown scenario information in the scenario data, accounting for the OWA required when analyzing test data from robotic systems

operating in complex contexts. We have not yet explained how we derive from the data $d = (\mathfrak{D}_1, \ldots, \mathfrak{D}_m)$ a scenario possibly containing unknown information, i.e., implementing a (partial) function $\Lambda_{T,V\cup\{*\}} : \bigcup_{h>0} \mathbb{D}^h \to (V \cup \{*\})^T$. Recall that this function assigns a given data trace its tag values. To implement $\Lambda_{T,V\cup\{*\}}$, the idea is to use formulae of a Boolean logic to decide the value of a tag for a given data trace. More specifically, we employ *two* formulae, one representing certainty about the tag value being 0, and the other formalizing the case where a tag value of 1 must hold. In the gray zone between both formulae (where none is true), the tag value $*$ is assigned to the data trace.

Formally, we assume to have some Boolean logic L with semantics $d \models \psi$ for $\psi \in \mathsf{L}$, indicating that formula ψ to hold given the data d. We can then define the tag value for a given data trace under an OWA via two formulae $\psi_t^+, \psi_t^- \in \mathsf{L}$ for safe satisfaction and violation of some tag $t \in T$, respectively:

$$\Lambda_{T,V\cup\{*\}}(d) \quad = \quad t \mapsto \begin{cases} 1, & \text{if } d \models \psi_t^+ \text{ and } d \not\models \psi_t^-, \\ 0, & \text{if } d \not\models \psi_t^+ \text{ and } d \models \psi_t^-, \\ *, & \text{if } d \not\models \psi_t^+ \text{ and } d \not\models \psi_t^-. \end{cases} \tag{2}$$

If both formulae hold, a modeling error is present and no value can be identified (hence the function being partial). If both formulae do not hold, no decision on the tag's value can be made and $*$ is assigned.

It remains to instantiate L, for which temporal logics lend themselves naturally due to their ability to reason over finite, discrete temporal data. They were successfully applied, albeit under a CWA [9,18,24,29], as also delineated in Sect. 6. In what follows, we show two such temporal logics, both based on Linear Temporal Logic over Finite Traces (LTL_f) [8]. For some language of atoms $\mathcal{A}$, $\mathrm{LTL}_\mathrm{f}^\mathcal{A}$ syntax is defined by the grammar $\psi ::= \mathcal{A} \mid \neg\psi \mid \psi \wedge \psi \mid \psi \,\mathcal{U}\, \psi \mid \bigcirc \psi$, where we allow for atoms from $\mathcal{A}$, the usual Boolean operators ($\wedge$ and $\neg$), as well as until ($\mathcal{U}$) and next ($\bigcirc$). $\mathcal{A}$ is intentionally left open: Whereas our first proposal assumes $\mathcal{A}$ as propositions, the second one uses model-theoretic semantics and defines $\mathcal{A}$ as Conjunctive Queries (CQs) over DLs.

4.1 Propositional Logic

In standard LTL_f [8], data are sequences of propositions $\mathcal{P}$, i.e., $\mathbb{D} = 2^\mathcal{P}$. A formula $\psi \in \mathrm{LTL}_\mathrm{f}^\mathcal{P}$ is satisfied by a data trace $d = (\mathfrak{D}_1, \ldots, \mathfrak{D}_m)$, written $d \models \psi$, if $d, 1 \models \psi$, where

$d, i \models P$, for $P \in \mathcal{P}$, iff $P \in \mathfrak{D}_i$,
$d, i \models \neg\psi$ iff $d, i \not\models \psi$,
$d, i \models \psi_1 \wedge \psi_2$ iff $d, i \models \psi_1$ and $d, i \models \psi_2$,
$d, i \models \bigcirc\, \psi$ iff $i < m$ and $d, i+1 \models \psi$, and
$d, i \models \psi_1 \,\mathcal{U}\, \psi_2$ iff ex. a $k \in [i, m]$ s.t. $d, k \models \psi_2$ and $d, j \models \psi_1$ for all $j \in [i, k)$.

Typical syntactical definitions apply, e.g., $\Diamond\ \psi = \mathsf{true}\, \mathcal{U}\, \psi$. As propositional logics are interpreted under a CWA, it remains to use ψ_t^+ and ψ_t^- for modeling

an OWA, i.e., by Eq. 2, the gray zone is present iff both conditions are false. For a meaningful OWA, users must carefully model the conditions ψ_t^+ and ψ_t^-.

Consider the example t_1 (whether a safety distance violation is present), for which we use, for the sake of the example, the half-speed rule as an over-approximation. To under-approximate, we use the distance requirements specified in for Automated Lane Keeping Systems in the UNECE Regulation No. 157 [33]. Both rules span the gray zone sketched in Fig. 2. With corresponding propositions, we can define inclusion and exclusion conditions for t_1:

- Inclusion ($\psi_{t_1}^+$): The requirements of the UNECE R157 were violated; $\Diamond\,(\texttt{lead\_vehicle} \wedge \neg \texttt{safety\_distance}_{\text{R157}})$.
- Exclusion ($\psi_{t_1}^-$): The half-speed rule was always adhered to; $\Box\,\neg(\texttt{lead\_vehicle} \wedge \neg \texttt{safety\_distance}_{\frac{1}{2}})$.

This approach adds the OWA only at the very end of the specification side (meaning the determination of the tag's value), while $\text{LTL}_\text{f}^{\mathcal{P}}$ formulae are interpreted under the CWA. This has the advantage of enabling efficient computation, as the word problem (deciding membership of a data trace to the language of a formula) in $\text{LTL}_\text{f}^{\mathcal{P}}$ can be decided in polynomial time [5], however, to the detriment of a fully integrated OWA. In our example, $\psi_{t_1}^+$ itself is interpreted under a CWA, i.e., if $\psi_{t_1}^+$ is not present in the data, $\neg\psi_{t_1}^+$ is assumed to hold.

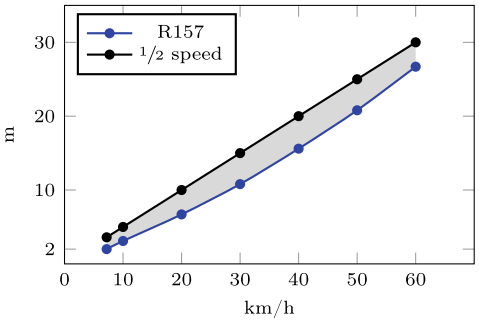


Fig. 2. Distance regulations.

4.2 Description Logics

DLs, a decidable fragment of first order logic, inherently make the OWA, contrary to propositional logic. They are a popular choice for augmenting data by background knowledge [3] which can then be queried by CQs. The temporal logic $\text{LTL}_\text{f}^{\text{CQ}}$ uses CQs as its atom language [39].

We model data over so-called individuals (observable entities) that can be classified by means of concepts and roles. For this, we assume name supplies $\mathsf{N_C}$ (concepts), $\mathsf{N_R}$ (roles), and $\mathsf{N_I}$ (individuals). Examples are `Walkway` or `Vehicle` as concepts and `v` as an individual. Data axioms are assignments of individuals to concepts and roles, and of the form $\texttt{C}(\texttt{i}_1)$ and $\texttt{r}(\texttt{i}_1, \texttt{i}_2)$ a concept $\texttt{C} \in \mathsf{N_C}$, role $\texttt{r} \in \mathsf{N_R}$, and individuals $\texttt{i}_1, \texttt{i}_2 \in \mathsf{N_I}$. For example, $\texttt{NonMovingVehicle}(\texttt{v})$ adds the individual `v` to the concept `NonMovingVehicle`. $\mathbb{D}$ is then the set of all data axiom sets, allowing to represent a finite data trace d as a sequence data axioms, i.e., $d = (\mathfrak{D}_1, \ldots, \mathfrak{D}_m)$ with $\mathfrak{D}_i \in \mathbb{D}$. Example temporal data are $d_{ex} = (\{\texttt{NonMovingVehicle}(\texttt{v}), \texttt{intersects}(\texttt{v}, \texttt{w}), \texttt{Walkway}(\texttt{w})\}, \emptyset)$, asserting `v` to be a non-moving vehicle intersecting a walkway first, followed by no observation.

The data are interpreted over an ontology $\mathcal{O}$, which is a set of axioms describing background knowledge of the form $C \sqsubseteq D$ (a concept D subsumes C), allowing to infer additional facts from the observed data. In the expressive DL $\mathcal{ALC}$, concepts can be composed of concept names ($\mathsf{N_C}$), intersections ($C \sqcap D$), unions ($C \sqcup D$), negations ($\neg C$), universal role restrictions ($\forall \mathtt{r}.C$), and existential role restrictions ($\exists \mathtt{r}.C$), for concepts C, D and a role $\mathtt{r} \in \mathsf{N_R}$. An example $\mathcal{ALC}$-ontology $\mathcal{O}_{ex} = \{\mathtt{NonMovingVehicle} \sqcap \exists \mathtt{interects}.\mathtt{ParkingSpot} \sqsubseteq \mathtt{ParkingVehicle}, \mathtt{Walkway} \sqsubseteq \mathtt{ParkingSpot}\}$ states that non-moving vehicles on parking spots, which include walkways, are parking.

DLs employ model-theoretic semantics by using temporal interpretations $\mathfrak{I}$, which are sequences $(\mathcal{I}_i)_{i \in \{1,\dots,m\}}$ of $\mathcal{ALC}$-interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Here, $\Delta^{\mathcal{I}}$ is a shared interpretation domain and $\cdot^{\mathcal{I}}$ maps concepts, roles, and individuals to its elements. Any model $\mathcal{I}$ of an $\mathcal{ALC}$-ontology and data axioms $(\mathcal{O}, \mathfrak{D})$ – we write $\mathcal{I} \models (\mathcal{O}, \mathfrak{D})$ – must respect its constraints. For brevity, we omit formal details on DL semantics here and refer the reader to the literature [3,39]. In our example, in any model $\mathcal{I}$ of $\mathcal{O}_{ex}$ it must hold that $\mathtt{Walkway}^{\mathcal{I}} \subseteq \mathtt{ParkingSpot}^{\mathcal{I}}$. Hence, for the ontology and data, it follows that in all its temporal models $\mathfrak{I}$, $\mathtt{v}$ must be parking in $\mathcal{I}_0$. No inferences on $\mathtt{v}$ can be made for the second timestamp (it might be parking or not parking), and it can belong to any concept in $\mathcal{I}_1$.

We now define $\mathrm{LTL}_{\mathrm{f}}^{\mathrm{CQ}}$, which incorporates DL semantics into LTL_f by using Boolean CQs as its atomic expressions $\mathcal{A}$ [39]. A Boolean CQ φ is a finite conjunction of data axioms, but additionally allows existentially quantified variables in place of individuals. We say that φ is entailed by some interpretation $\mathcal{I}$ ($\mathcal{I} \models \varphi$) if the query's constraints are reflected in the interpretation under a suitable mapping of the variables into the interpretation domain. For example, the Boolean CQ $\exists x.\mathtt{ParkingVehicle}(x)$ is entailed by any model $\mathcal{I}$ of $(\mathcal{O}_{ex}, (d_{ex})_1)$, as x can be assigned $\mathtt{v}^{\mathcal{I}}$. We lift entailment from CQs to $\mathrm{LTL}_{\mathrm{f}}^{\mathrm{CQ}}$ formulae ψ:

$\mathfrak{I}, i \models \varphi$ iff $\mathcal{I}_i \models \varphi$ (for a Boolean CQ φ),
$\mathfrak{I}, i \models \neg\psi$ iff $\mathfrak{I}, i \not\models \psi$,
$\mathfrak{I}, i \models \psi_1 \wedge \psi_2$ iff $\mathfrak{I}, i \models \psi_1$ and $\mathfrak{I}, i \models \psi_2$,
$\mathfrak{I}, i \models \bigcirc \psi$ iff $i < m$ and $\mathfrak{I}, i+1 \models \psi$, and
$\mathfrak{I}, i \models \psi_1 \,\mathcal{U}\, \psi_2$ iff ex. a $k \in [i, m]$ s.t. $\mathfrak{I}, k \models \psi_2$ and $d, j \models \psi_1$ for all $j \in [i, k)$.

Then, ψ is entailed by an ontology and data trace $(\mathcal{O}, d)$, written $(\mathcal{O}, d) \models \psi$, if for *all* their models $\mathfrak{I}$ it holds that $\mathfrak{I}, 1 \models \psi$.

$\mathrm{LTL}_{\mathrm{f}}^{\mathrm{CQ}}$ makes the OWA through this model-based semantics. Specifically, from the non-entailment of a formula its negation does not follow: By definition, the negated universal quantification over all interpretations (non-entailment) implies solely the existence of a model consistent with the negated fact (satisfiability of the negation), which is not equivalent to entailment of the negation. For example, $(\mathcal{O}_{ex}, (d_{ex})_2) \not\models \mathtt{ParkingVehicle}(\mathtt{v})$ ($\mathtt{v}$ is not guaranteed to be parking in the second timestamp) does not imply $(\mathcal{O}, (d_{ex})_2) \models \neg\mathtt{ParkingVehicle}(\mathtt{v})$ ($\mathtt{v}$ is not parking); only in some but not all models $(\neg\mathtt{ParkingVehicle})(\mathtt{v})$ holds.

Similar to what is done for $\mathrm{LTL}_{\mathrm{f}}^{\mathcal{P}}$ above, we define a tag $t \in T$ via formulae $\psi_t^+, \psi_t^- \in \mathrm{LTL}_{\mathrm{f}}^{\mathrm{CQ}}$. In the simplest case, $\psi_t^- = \neg\psi_t^+$, as, under the OWA

semantics, both formulae might be not entailed, enabling the desired gray zone. The example tag t_3 is a suitable candidate for formalization via DLs, since engineers can model their ontological knowledge on what encompasses parking vehicles, as exemplified by the $\mathcal{ALC}$-ontology $\mathcal{O}_{ex}$ above. A more involved ontology on parking vehicles is given in the reproducibility artifact [31]. We can then ask if at some point a parking vehicle $\mathtt{v}$ was passed by the system $\mathtt{s}$: $\psi^+_{t_3} = \Box\, \mathtt{ParkingVehicle}(\mathtt{v}) \wedge \Diamond\, \big(\mathtt{inProximity}(\mathtt{s},\mathtt{v}) \wedge \mathtt{inFront}(\mathtt{v},\mathtt{s}) \wedge \Diamond\, (\mathtt{inProximity}(\mathtt{s},\mathtt{v}) \wedge \mathtt{behind}(\mathtt{v},\mathtt{s}))\big)$, and set $\psi^-_{t_3} = \neg\psi^+_{t_3}$. If $\mathtt{v}$ is not explicitly modeled as parking in the data, we require interpreting d_{ex} under $\mathcal{O}_{ex}$ for additional inferences. While we can infer $\mathtt{v}$ to be parking for the first timestamp in d_{ex}, its second timestamp does not allow making any inferences in that regard. Hence, both ψ_{t_3} and $\neg\psi_{t_3}$ are not entailed on $(\mathcal{O}_{ex}, d_{ex})$, resulting in the tag value being $*$.

To summarize, DLs use the OWA *inherently* for reasoning over *all* axioms, adding gray zones also to all intermediate reasoning steps, e.g., determining whether $\mathtt{ParkingVehicle}(\mathtt{v})$ holds. This comes to the expense of complexity, as, in contrast to $\mathrm{LTL}_\mathrm{f}^{\mathcal{P}}$, the word problem of $\mathrm{LTL}_\mathrm{f}^{\mathrm{CQ}}$ is ExpTime-complete [39].

5 Evaluation

We implemented the algorithms of Sect. 3.2 for computing both bounds into STARS, a scenario coverage measurement framework [29]. For calculating MinCover, we use the Z3 solver[1]. For MaxCover, we allow using either Edmonds' blossom or the Hopcroft-Karp algorithm for maximum cardinality (bipartite) matching, as implemented in jgrapht[2].

For computing tag values for a given data trace, STARS provides, by default, CMFTBL, which can be seen as an extension of $\mathrm{LTL}_\mathrm{f}^{\mathcal{P}}$. Equation 2 allows users of STARS to specify two formulae for safe inclusion and exclusion of a tag. Additionally, we added a module to STARS that includes Topllet [40], a software that implements $\mathrm{LTL}_\mathrm{f}^{\mathrm{CQ}}$ [37,39]. The module acts as a drop-in replacement for CMFTBL [36]. In STARS, we allow tags to be interpreted either under the CWA or OWA, justified by the observation that information and specification can be sometimes assumed to be virtually perfect. We use our implementation for answering the following main research questions:

Q1 What are grounded assumptions on the distribution of $*$-entries in real data?
Q2 Do the lower and upper bound provide useful approximations on coverage under such $*$-distributions?
Q3 Can both bounds be computed efficiently also on practically relevant data configurations, and how is performance impacted by different factors?

[1] https://github.com/Z3Prover/z3.
[2] https://jgrapht.org.

5.1 Tag Value Distributions on Real-World Data

For answering Q1, we again rely on the three example OWA tags t_1 to t_3. The first tag ("safety distance") can be implemented in a propositional fashion (cf. Sect. 4.1). Similarly, corresponding OWA predicates for t_2 ("presence of a pedestrian or bicycle") can be defined: We degrade any perceived bicycle with a speed over 12 m/s and pedestrian over 5 m/s as a not further identifiable vulnerable road user (there is a realistic chance that the bicycle was, in fact, a motorcycle, and the pedestrian was actually mounted on a vehicle), and assign $*$ in those cases. The third tag ("passing parking vehicle") lends itself for formalization in DLs, as seen in Sect. 4.2, for which we specified a corresponding query in the input language of Topllet.

For real-world data, we use the established inD data set, which contains roughly 7 500 scenarios from drone camera recordings of four intersections in Aachen, Germany [4]. The videos are post-processed by neural networks to extract trajectories and classify traffic participants, and also contain road network information. Each scenario represents one ego vehicle traversing an intersection. Based on these details, we determine values for our three example OWA tags t_1 to t_3. Since the drone data does not contain scenarios at low sun positions, the example tag t_4 was excluded from the analysis. The evaluation scripts are given in the reproduction artifact; unfortunately, an individual copy of the data must be obtained due to licensing.

Table 1 shows the resulting distributions of tag values. Whereas the first two tags, relating to safety distance fuzziness and vague perception classification, exhibit less than 10% $*$-entries, the third tag accumulates to roughly 14%. For t_3, it is certainly possible to reduce uncertainty by a more refined specification of parking vehicles. We draw upon these empirical observations to determine grounded probabilities for $*$-entries in the subsequently used synthetic data.

Table 1. Distributions for the OWA example tags on inD.

V_t \ t	t_1	t_2	t_3
0	73.03%	14.91%	46.43%
1	17.60%	78.95%	39.59%
$*$	9.37%	6.14%	13.98%

5.2 Quality and Runtime Analysis on Synthetic Data

For the remaining research questions, we require controlled data to assess quality and runtime under different configurations. This includes the numbers of tags, the split between OWA and CWA tags, and the distribution of $*$-entries. This level of control is unfortunately not present in real-world data, which is why we construct synthetic data for answering Q2 and Q3. The data are generated for all numbers $n_t \in [1, 15]$ of tags, where the number of OWA tags is $n_{\text{owa}} \in [1, n]$ and the number of CWA tags is $n_{\text{cwa}} = n_t - n_{\text{owa}}$, with different probabilities of $*$-entries $p_* \in \{5\%, 10\%, 15\%, 20\%\}$, derived from our practical experiences with inD. The data were created by first generating a "ground truth", i.e., records where each value is known, which are then made unknown by probabilistically

overwriting entries with $*$-values. This allows for reporting "real" coverage values later on. The probability of all tags being either true or false is set to 50% each. We generate data in batches of scenarios and then compute coverage for all data. For efficiency reasons, we increase the batch size depending on the numbers of assessed tags to a maximum of 10 000 for $n_t \geq 13$. All subsequently presented computations were done single threaded on a `Intel Xeon Gold 6342 CPU` running Ubuntu 20 using docker. While the data were generated probabilistically, practical experience showed that results for different seeds did not differ meaningfully, and hence only a single run is executed per configuration. The results of all configurations are available in our replication artifact [31].

Coverage Approximation Quality. For Q2, we start by showcasing the progression of MaxCover, MinCover, and real coverage on the configuration $n_{\text{owa}} = 5$, $n_{\text{cwa}} = 10$, and $p_* = 10\%$ in Fig. 3. We additionally depict naïve lower and upper bounds. The upper bound is constructed by assuming for every $*$ that both options happened (which is impossible), and the lower bound by ignoring all scenarios containing $*$. Observe that the x-axis of Fig. 3 is scaled logarithmically, since coverage improvements degrade exponentially over time.

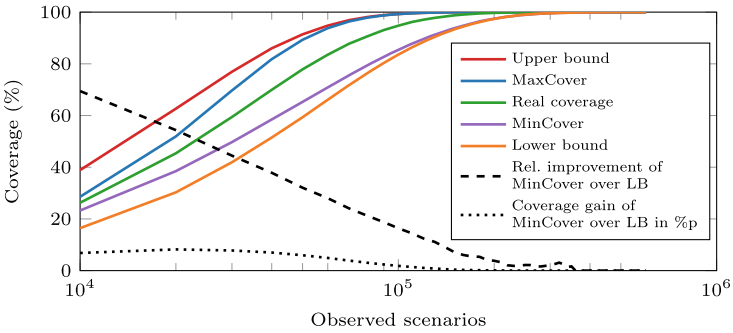


Fig. 3. Coverage and derived bounds for $n_{\text{owa}} = 5, n_{\text{cwa}} = 10, p_* = 10\%$.

In Fig. 3, we focus only on the performance gain of MinCover, due to its higher practical relevancy (as argued in Sect. 3) and for ease of presentation. Naturally, improvements are most immanent at the beginning, with up to 69% relative and 7% points absolute gain. For full coverage, all bounds nearly coincide – however, during short development cycles observed in practice, full coverage is likely an unrealistic goal. Rather, 80% appears as a more reasonable target in industrial settings, for which MinCover still reduces the error of the naïve lower bound to real coverage by 28% on the example. Over all conducted experiments with $n_t \geq 5$, we report a reduced error of MinCover over the lower bound of $30.61\% \pm 6.23\%$ at 80% real coverage. For the upper bound, error reduction becomes uninformative at 80% (since it approaches zero). We hence report it at 50% real coverage, where we achieve an error reduction of $35.73\% \pm 18.76\%$.

Runtime Analysis. In Q3 we ask whether coverage approximations can be computed on real-world settings. Figure 4 reports the gap between MinCover and MaxCover for $n_{\text{owa}} = 10$ and $n_{\text{cwa}} \in [0, 8]$ at $p_* = 5\%$ on a normalized scale (with 1 representing the sample at which the gap reached zero). Evidently, there is no influence of additional closed tags on the spread of the bounds, so we conduct this analysis on OWA tags only. For MaxCover, Edmonds' blossom and the Hopcroft-Karp algorithm yield comparably fast computation times (with the latter being slightly advantageous). We however omit those in the runtime analysis, as MinCover uses on average $> 98\%$ of the total runtime for $n_{\text{owa}} \geq 2$.

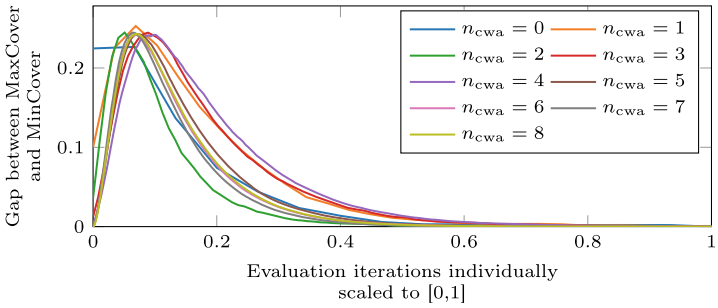


Fig. 4. Bound gap at $n_{\text{owa}} = 10$, $p_* = 5\%$ over added CWA tags n_{cwa}.

Figure 5 reports the runtime of MinCover for $n_{owa} = 13$ and $n_{cwa} = 0$ with varying p_*, where each point represents one computation of MinCover on the full dataset up the given amount of scenarios. The plot indicates a sublinear relationship between the number of scenarios and the required computation time, with spikes at the beginning most likely being due to solver initialization. Lower p_*-values reach 100% MinCover after fewer scenarios (due to lesser degrees of freedom in selecting a worst case), and their graphs therefore abort early in Fig. 5. This shows that coverage bounds can be computed timely even at realistic data sizes of 10^6 (recall that inD contains less than 10^4 scenarios). To assess the influence of different numbers of OWA tags on runtime, Fig. 6 compares the wall-clock time of MinCover for the last iteration, i.e., when the coverage hit 100%, of all examined configurations, to the amount of observed scenarios needed to achieve 100% coverage. The computations for $n_{\text{owa}} > 13$ at $p_* = 20\%$ and $n_{\text{owa}} > 15$ had to be terminated due to excessive runtime.

Our results show that for up to but excluding 15 OWA tags, MinCover can be successfully computed in 35 s or less at the last iteration (meaning on as many scenarios as required for 100% coverage). We also observe a direct correspondence of runtime to the number of scenarios required to obtain full coverage. Most likely, this behavior originates from adding for each new scenario its explicated options as a new hard clause. The solver must hence consider, at worst, 3^{n_t} different hard clauses. We remark that also adding CWA tags influences the overall runtime of the solver exponentially: For example, at

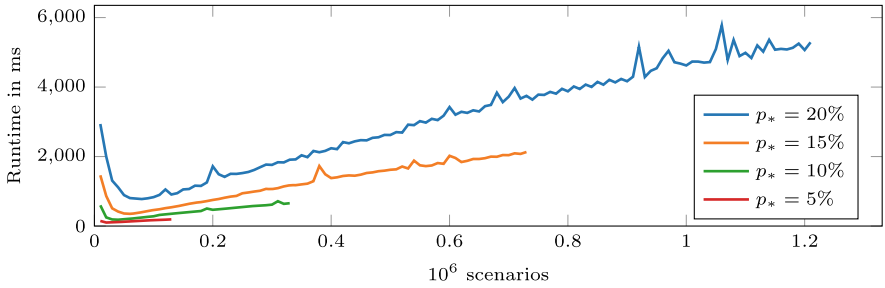
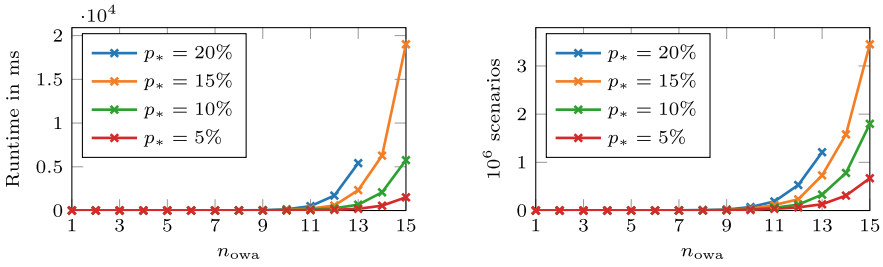


Fig. 5. Runtime of MinCover for $n_{\text{owa}} = 13, n_{cwa} = 0$ over the data size.



(a) Runtime

(b) Scenarios needed for 100% coverage

Fig. 6. Comparison of wall-clock time to compute last iteration of MinCover, and scenarios needed for 100% coverage across different amounts of OWA tags.

$n_{\text{cwa}} = 9, n_{\text{owa}} = 10, p_* = 5\%$, the calculation was terminated after exceeding 32 h wall-clock time. In practice, the number of tags and their definitions must thus be chosen carefully.

6 Related Work

We have above assumed that an input classification system is available for computing coverage. It must be methodically derived and specified, e.g., based on an ontology [14,38] or combinatorics [35]. Here, the general assumption is that the system shows equivalent behavior w.r.t. safety within a class [1]. Subsequently, recognizing the specified classes in data is required, e.g., by using classifiers trained on human reference behavior [28], trajectory matching [35], or comparing against sequences of events [16]. Another popular choice in automated driving are temporal logics, e.g., MTL or STL over discrete-time [18,24], LTL [9], and CMFTBL [29], which we also used as a foundation. These formalisms originally all make the CWA. This assumption has been extended in this work.

Once scenario classes are specified and recognized in test data, coverage can be computed. In software engineering, coverage (such as branch, statement, and MC/DC coverage) is well-established [43]. Transfer to the emerging domain of

automated systems is ongoing [13], where it forms one of the fundamental safety challenges [25,27]. While arguments have been made against excessive dogmatic use of coverage in software engineering [12,17], it may be justified as an important component in testing automated systems if metrics exhibit certain qualities, such as tractability [1]. One of the most foundational metrics for automated robotic systems is tag-based coverage [13,29] – which has also been used in this paper as a canonical basis for demonstrating how the OWA can generally be integrated into coverage. Other metrics exist, including tree-based coverage (removing implausible combinations by constraints [29]), time-based coverage (the duration where no class applies [13]), probabilistically-bounded coverage (counting only combinations that are at least somewhat likely [1]), t-wise coverage (restricting to t-combinations, where the assumption is that many hazards are triggered by at most t classes [32]), quantitative k-projection coverage (a weighted, rule-based extension of t-wise coverage [6]), or concrete scenario coverage (where classification is omitted altogether [2]). Again, all these works make the CWA. We are not aware of any OWA work in our context. Thus, to the best of knowledge, our presented formalism is the first OWA framework in the field of test coverage.

7 Conclusion

In this work, we extended coverage from closed to open world environments, under which MinCover and MaxCover give lower and upper approximations on the actual coverage. Our novel formal framework enabled theoretical complexity analyses, where the corresponding decision problem is D^p-complete for MinCover and polynomially decidable for MaxCover. We presented two ways of relying on temporal logics for an open world scenario classification, one based on propositional and the other on DL semantics. Experiments with an implementation of our formal framework – based on MaxSAT and maximum cardinality matching algorithms – showed both bounds to be feasibly computable on realistic data, and also add value over naïve lower and upper bound computations.

Future work includes optimizing the MinCover computation, e.g., by catching certain cases to avoid running the solver, since our results showed that most of the time is spent solving MaxSAT. Additionally, we plan to assess real-world data from an ego-perspective to also include occlusion effects, which are a relevant source of unknowns not present in inD. To summarize, with this work, we aim at sparking interest in a challenging new field in test coverage analysis, motivated from practical demands while simultaneously offering surprising theoretical subtleties.

References

1. Alexander, R., Hawkins, H.R., Rae, A.J.: Situation coverage - a coverage criterion for testing autonomous robots. Report, The University of York (2015). https://eprints.whiterose.ac.uk/id/eprint/88736/

2. Amersbach, C., Winner, H.: Defining required and feasible test coverage for scenario-based validation of highly automated vehicles. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 425–430. IEEE (2019). https://doi.org/10.1109/ITSC.2019.8917534
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications, 2 edn. Cambridge University Press (2007). https://doi.org/10.1017/cbo9780511711787
4. Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., Eckstein, L.: The inD dataset: a drone dataset of naturalistic road user trajectories at German intersections. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 1929–1934. IEEE (2020). https://doi.org/10.1109/iv47402.2020.9304839
5. Bundala, D., Ouaknine, J.: On the complexity of temporal-logic path checking. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8573, pp. 86–97. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43951-7_8
6. Cheng, C.-H., Huang, C.-H., Yasuoka, H.: Quantitative projection coverage for testing ML-enabled autonomous systems. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 126–142. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_8
7. Chuzhoy, J., Khanna, S.: A faster combinatorial algorithm for maximum bipartite matching. In: Proceedings of the 2024 Annual ACM-Siam Symposium on Discrete Algorithms (SODA), pp. 2185–2235. Society for Industrial and Applied Mathematics (2024). https://doi.org/10.1137/1.9781611977912.79
8. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI 2013, pp. 854–860. Association for Computing Machinery, New York, NY, USA (2013). https://dl.acm.org/doi/abs/10.5555/2540128.2540252
9. Esterle, K., Gressenbuch, L., Knoll, A.: Formalizing traffic rules for machine interpretability. In: 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS), pp. 1–7. IEEE (2020). https://doi.org/10.1109/CAVS51000.2020.9334599
10. Federal Ministry for Digital and Transport (Germany): Straßenverkehrs-Ordnung (StVO). Regulation BGBl. I S. 367, BMDV (2013). https://www.gesetze-im-internet.de/stvo_2013/
11. Federal Ministry for Digital and Transport (Germany): Verordnung zur Regelung des Betriebs von Kraftfahrzeugen mit automatisierter und autonomer Fahrfunktion und zur Änderung straßenverkehrsrechtlicher Vorschriften. Regulation 86/22, BMDV (2022). https://www.bundesrat.de/bv.html?id=0086-22
12. Gay, G., Staats, M., Whalen, M.W., Heimdahl, M.P.E.: Moving the goalposts: coverage satisfaction is not enough. In: Proceedings of the 7th International Workshop on Search-Based Software Testing (SBST), SBST 2014, pp. 19–22. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2593833.2593837
13. de Gelder, E., Buermann, M., ob den Camp, O.: Coverage metrics for a scenario database for the scenario-based assessment of automated driving systems. In: 2024 IEEE International Automated Vehicle Validation Conference (IAVVC), pp. 1–8. IEEE (2024). https://doi.org/10.1109/iavvc63304.2024.10786405
14. de Gelder, E., et al.: Towards an ontology for scenario definition for the assessment of automated vehicles: an object-oriented framework. IEEE Trans. Intell. Veh. **7**(2), 300–314 (2022). https://doi.org/10.1109/tiv.2022.3144803

15. de Gelder, E., Singh, T., Hadj-Selem, F., Vidal Bazan, S., Op den Camp, O.: Scenario metrics for the safety assurance framework of automated vehicles: a review of its application. Vehicles **7**(3), 100 (2025). https://doi.org/10.3390/vehicles7030100
16. de Gelder, E., Manders, J., Grappiolo, C., Paardekooper, J.P., ob den Camp, O., de Schutter, B.: Real-world scenario mining for the assessment of automated vehicles. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–8. IEEE (2020). https://doi.org/10.1109/itsc45102.2020.9294652
17. Hamlet, D., Taylor, R.: Partition testing does not inspire confidence (program testing). IEEE Trans. Software Eng. **16**(12), 1402–1411 (1990). https://doi.org/10.1109/32.62448
18. Hekmatnejad, M., et al.: Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In: Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2019, pp. 1–11. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3359986.3361203
19. Hopcroft, J.E., Karp, R.M.: A N5/2 algorithm for maximum matchings in bipartite. In: 12th Annual Symposium on Switching and Automata Theory (SWAT 1971). IEEE (1971). https://doi.org/10.1109/swat.1971.1
20. International Organization for Standardization: Road vehicles — safety of the intended functionality. ISO Standard 21448:2022, ISO (2022). https://www.iso.org/standard/77490.html
21. International Organization for Standardization: Road vehicles — test scenarios for automated driving systems — scenario categorization. ISO Standard 34504:2024, ISO (2024). https://www.iso.org/standard/78953.html
22. Keet, C.M.: Open World Assumption, p. 1567. Springer New York, New York, NY (2013). https://doi.org/10.1007/978-1-4419-9863-7_734
23. Krentel, M.W.: The complexity of optimization problems. J. Comput. Syst. Sci. **36**(3), 490–509 (1988). https://doi.org/10.1016/0022-0000(88)90039-6
24. Maierhofer, S., Moosbrugger, P., Althoff, M.: Formalization of intersection traffic rules in temporal logic. In: 2022 IEEE Intelligent Vehicles Symposium (IV), pp. 1135–1144. IEEE (2022). https://doi.org/10.1109/IV51971.2022.9827153
25. Neurohr, C., Westhofen, L., Henning, T., de Graaff, T., Mohlmann, E., Bode, E.: Fundamental considerations around scenario-based testing for automated driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 121–127. IEEE (2020). https://doi.org/10.1109/iv47402.2020.9304823
26. Papadimitriou, C.H., Yannakakis, M.: The complexity of facets (and some facets of complexity). In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing - STOCK 1982, STOC 1982, pp. 255–260. Association for Computing Machinery, New York, NY, USA (1982). https://doi.org/10.1145/800070.802199
27. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE Access **8**, 87456–87477 (2020). https://doi.org/10.1109/ACCESS.2020.2993730
28. Roesener, C., Fahrenkrog, F., Uhlig, A., Eckstein, L.: A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1360–1365. IEEE (2016). https://doi.org/10.1109/itsc.2016.7795734
29. Schallau, T., Naujokat, S., Kullmann, F., Howar, F.: Tree-based scenario classification. In: Benz, N., Gopinath, D., Shi, N. (eds.) NASA Formal Methods. LNCS,

vol. 14627, pp. 259–278. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-60698-4_15
30. Schmid, D., et al.: A qualitative analysis of the german road traffic regulations (StVO) for automated driving. In: 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), pp. 1–8. IEEE (2025). https://doi.org/10.1109/IAVVC61942.2025.11219635
31. Schmid, D., Westhofen, L., Schallau, T.: Data from: test coverage of automated robotic systems in open world environments (2026). https://doi.org/10.5281/zenodo.17775279. Dataset
32. Sun, J., Zhang, H., Zhou, H., Yu, R., Tian, Y.: Scenario-based test automation for highly automated vehicles: a review and paving the way for systematic safety assurance. IEEE Trans. Intell. Transp. Syst. **23**(9), 14088–14103 (2022). https://doi.org/10.1109/TITS.2021.3136353
33. United Nations: Uniform provisions concerning the approval of vehicles with regard to automated lane keeping systems (ALKS). UN Regulation 157, United Nations (2021). https://documents.un.org/symbol-explorer?s=E/ECE/TRANS/505/REV.3/ADD.156, e/ECE/TRANS/505/Rev.3/Add.156
34. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing - STOC 1982, STOC 1982, pp. 137–146. Association for Computing Machinery, New York, NY, USA (1982). https://doi.org/10.1145/800070.802186
35. Weber, H., Glasmacher, C., Schuldes, M., Wagener, N., Eckstein, L.: Holistic driving scenario concept for urban traffic. In: 2023 IEEE Intelligent Vehicles Symposium (IV). IEEE (2023). https://doi.org/10.1109/iv55152.2023.10186385
36. Westhofen, L.: STARS logic MTCQ (2025). https://doi.org/10.5281/zenodo.17791768. Software
37. Westhofen, L., Jung, J.C., Neider, D.: Temporal conjunctive query answering via rewriting. Proc. AAAI Conf. Artif. Intell. **39**(14), 15221–15229 (2025). https://doi.org/10.1609/aaai.v39i14.33670
38. Westhofen, L., Neurohr, C., Butz, M., Scholtes, M., Schuldes, M.: Using ontologies for the formalization and recognition of criticality for automated driving. IEEE Open J. Intell. Transp. Syst. **3**, 519–538 (2022). https://doi.org/10.1109/ojits.2022.3187247
39. Westhofen, L., Neurohr, C., Jung, J.C., Neider, D.: Answering temporal conjunctive queries over description logic ontologies for situation recognition in complex operational domains. In: Finkbeiner, B., Kovács, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. LNCS, pp. 167–187. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-57246-3_10
40. Westhofen, L., Neurohr, C., Jung, J.C., Neider, D.: Topllet: an optimized engine for answering metric temporal conjunctive queries. In: NASA Formal Methods, pp. 314–321. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-60698-4_18
41. Westhofen, L., Schallau, T., Schmid, D., Naujokat, S., Howar, F., Neider, D.: Supplementary material to 'test coverage of automated robotic systems in open world environments'. Supplementary Mater. (2026). https://doi.org/10.5281/zenodo.18803837

42. Xiao, M.: An exact MaxSAT algorithm: further observations and further improvements. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22), IJCAI-2022, pp. 1887–1893. International Joint Conferences on Artificial Intelligence Organization (2022). https://doi.org/10.24963/ijcai.2022/262
43. Zhu, H., Hall, P.A.V., May, J.H.R.: Software unit test coverage and adequacy. ACM Comput. Surv. **29**(4), 366–427 (1997). https://doi.org/10.1145/267580.267590