Synthesizing Traffic Scenarios from Formal Specifications Using Reachability Analysis

Florian Finkeldei and Matthias Althoff

***Abstract*— Scenario-based testing is a promising approach for the verification of automated vehicles (AVs). In this paper, we present a novel approach that combines reachability analysis and numerical optimization to derive concrete scenarios from formal specifications. This promises to address multiple deficiencies of previous approaches: Improvement of computation times, handling of nonlinear specifications (e.g., traffic rules), and incorporation of criticality metrics. Our evaluation shows that the computation time increases linearly with the number of agents and the time horizon, compared to a typically exponential increase for methods without reachability analysis.**

I. Introduction

Autonomous driving has enormous potential for improving safety, conservation of resources, and productivity gains [1], [2]. A key challenge for the homologation and societal adoption of AVs is to guarantee their functional safety [3]. One promising approach is scenario-based testing, in which the behavior of AVs is simulated in relevant scenarios [4]. A scenario contains information on the road infrastructure as well as the abstract behavior of agents [5]. Menzel et al. [6] distinguish between *functional*, *logical*, and *concrete* scenarios, with concrete ones offering the highest level of detail [4]. The presented traffic scenario synthesis derives concrete scenarios from logical ones. Existing approaches are presented subsequently.

A. Related Work

We distinguish between knowledge-based and data-driven generation of concrete scenarios [3], [4], [7]. Data-driven approaches rely on a database of recorded real-world traffic scenarios [7, Sec. III]. This enables data-driven approaches to accurately reflect realistic human behavior [7, Sec. VII-A]. Clustering techniques are often used to extract relevant recordings [4]. In addition, machine learning approaches can generate new scenarios based on previously learned behavior from datasets [4], [7], [8]. As situations that require significant actions arise only rarely in real-world traffic, data-driven approaches have the disadvantage that their collection and selection process is inefficient [7, Sec. VII-B]. In addition, the desired behavior of the generated scenario cannot be properly controlled [7, Sec. VII-E].

Knowledge-based approaches instead leverage external knowledge, such as traffic rules, physical laws, or formal specifications, to guide the generation of concrete scenarios [7, Sec. V]. As such, they are capable of testing AVs against formal requirements, see [9]. However, depending on which type of knowledge is used as input, the behavior of the traffic participants might be unrealistic [7, Sec. VII-A]. Subsequently, we categorize knowledge-based approaches into those using ontologies or temporal logics.

Bagschik et al. [10] use ontologies for the knowledge description of traffic scenarios. One straightforward approach to derive concrete scenarios is to sample values for each parameter of a scenario directly from the permissible range defined in the logical description [6]. Variation methods aim at selecting the samples more sensibly [3], e.g., with boundary value analysis [11] or statistical methods [12]. Other approaches for converting logical scenarios into concrete ones derive a valid combination of specifications from the ontology description and use these as the input to traffic simulators, see [13], [14], [15].

Besides ontologies, expert knowledge, traffic rules, and scenario specifications can be described by temporal logic. In the control engineering domain, several works exist that solve optimal control problems considering temporal logics, see [16], [17]. This is described in more detail in [18], where the formal (logical) scenario description is converted into a mixed integer quadratic programming (MIQP) optimization. While the quality of the generated trajectories is satisfying, the runtime of MIQP increases exponentially with the number of binary variables [19]. Thus, the work in [20] reduces the number of binary variables originating from temporal logic. Another limitation of the MIQP optimization approach is that some traffic rules cannot be directly formulated as linear constraints, such as the evaluation of the safe distance predicate [21]. Moreover, for efficient gradient-based numerical optimization strategies to work, the objective function must be smooth [22]. When using criticality metrics as the objective function, this is not generally applicable [23], [24].

B. Contributions and Structure

We present a novel knowledge-based approach to synthesizing traffic scenarios, which overcomes the aforementioned disadvantages of MIQP approaches. To this end, we leverage the computation of reachable sets that are a) specification-compliant, b) forward-consistent, and c) convex, making it possible to find the optimal trajectories within them using quadratic programming (QP). The mediation process among the agents is handled by the reachable analysis instead of the previously necessary mixed integer formalism. As a consequence, a) the computational performance is substantially

* This work was supported by the German Research Foundation (DFG), grant AL 1185/17-1, and the Horizon Europe program, grant 101076165 (i4Driving).

All authors are with the School of Computation, Information, and Technology at the Technical University of Munich, 85748 Garching, Germany. florian.finkeldei@tum.de, althoff@in.tum.de

improved, b) traffic rules can be considered, and c) criticality metrics can be included in objective functions.

In the following section, preliminaries are introduced. The methodology of our novel approach is presented in Sec. III. In Sec. IV, its performance is assessed and compared to the work of Klischat et al. [18] based on two numerical examples. Finally, the paper concludes with a summary and an outlook on future work.

II. Preliminaries

To describe traffic scenarios, we formalize the road infrastructure and agents, including their coordinate frame, system dynamics, and abstract behavior specification.

A. Road Infrastructure

The road infrastructure description follows the CommonRoad scenario specification. CommonRoad scenarios consist of a lanelet network, intersections, traffic signs and lights, static obstacles, etc. [25]. The backbone of CommonRoad scenarios are lanelets, which are defined by polylines of their left and right boundaries (see Fig. 1). They also contain references to their predecessors, successors, and adjacent lanelets. In addition to the information provided by CommonRoad scenario files, information facilitating the subsequent tasks is provided: For each lanelet, its centerline and references to merging and diverging lanelets are stored. Furthermore, the conflict sections of lanelets at intersections and merges (see Fig. 2) are determined similarly to [18, Sec. II-C].

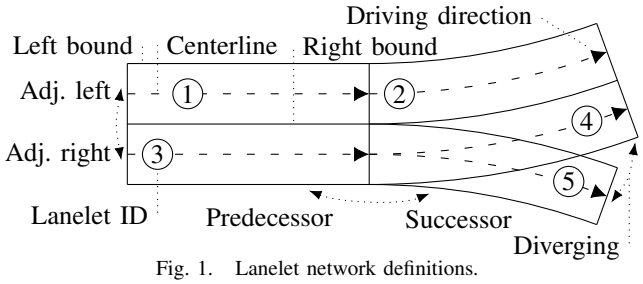

Fig. 1. Lanelet network definitions.

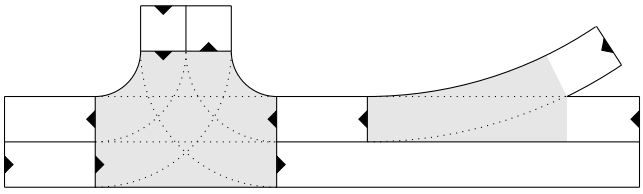

Fig. 2. Conflict sections (grey) at an intersection (left) and a merge (right).

B. Agents

To reference agents, an identifier $\square^i$ is used. For convenience, we define a shorthand notation

$$A^{\{1,2,...,\mathrm{N}\}} \Leftrightarrow \{A^i \mid i \in \{1,2,...,\mathrm{N}\}\}, \tag{1}$$

with N being the number of agents.

1) Route-based Coordinate Frame: Each agent A^i is initialized with a route γ^i. It is defined as a sequence of lanelets $(l_1^i, l_2^i, ..., l_{\mathrm{R}_i}^i)$ that the agent A^i traverses over the course of time. Based on the route γ^i, a reference path Γ^i is composed for each agent A^i similar to the procedure of the CommonRoad route planner[1].

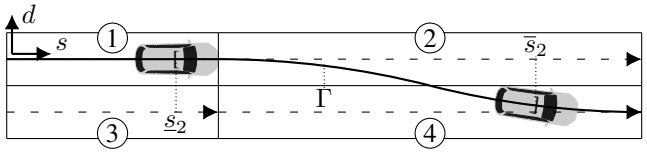

Fig. 3. Reference path Γ for route $\gamma = (1, 2, 4)$ and lonigtudinal position interval $[\underline{s}_2, \overline{s}_2]$ in which the agent intersects with lanelet l_2.

With the reference path Γ serving as a curvilinear coordinate frame, the longitudinal position s and lateral position d of an agent A are defined as indicated in Fig. 3. For each lanelet $l_r \in \gamma$, the longitudinal position interval $[\underline{s}_r, \overline{s}_r]$ in which the agent A intersects the lanelet l_r is determined (see Fig. 3). The symbols $\underline{\square}$ and $\overline{\square}$ indicate the lower and upper limits, respectively. Analogously, the longitudinal position intervals of the agents intersecting with conflict sections are computed. Preprocessing and storing this information facilitates the evaluation of predicates (see Sec. III-B.2). This procedure is similar to [18, Sec. II-C].

2) Interference of Agents: Next, the relationship among the agents is analyzed. For those agent pairs (A^{i_1}, A^{i_2}) whose reference paths $(\Gamma^{i_1}, \Gamma^{i_2})$ intersect, the intersection point is determined in their respective curvilinear coordinate frames. This will be relevant for the evaluation of some predicates (see Sec. III-B.2). This procedure is identical to [18, Sec. II-B].

3) System Dynamics: A linear point mass model with acceleration as an input variable is used for each agent. Both acceleration and velocity are constrained by upper and lower bounds. In this paper, we only consider longitudinal dynamics. The state $x^i = [s, v]^\mathsf{T}$ of an agent A^i is composed of its longitudinal position s and velocity v. Accordingly, the input $u^i = a^i$ is the longitudinal acceleration a^i. We use a time discretization with a fixed time increment Δt, and the acceleration a^i is assumed to be constant during each time step $\square_k$. Thus, the discrete-time vehicle model describing the longitudinal system dynamics is

$$x_{k+1}^i = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{A_\mathrm{s}} x_k^i + \underbrace{\begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}}_{B_\mathrm{s}} u_k^i, \tag{2}$$

with bounded acceleration $a_k^i \in [\underline{a}^i, \overline{a}^i] =: \mathcal{U}^i$, and velocity $v_k^i \in [\underline{v}^i, \overline{v}^i]$. In addition, the initial state of each agent A^i is limited to the convex set $\mathcal{R}_0^i$. The matrices A_s and B_s are the system and input matrices, respectively.

4) Abstract Behavior Specification: For the abstract specification of the behavior of each agent and the formalization of traffic rules [21], we use a subset of metric temporal logic (MTL) [26]. Its clarity and flexibility make MTL well suited for this task. MTL consists of temporal and logic operators that connect predicates [26].

Currently, our implementation supports the logic connective *and* as well as the temporal *always* operation. These

[1] commonroad.in.tum.de/tools/route-planner

offer the same functionality as previous MIQP approaches [18]. Other logic connectives and temporal operators (e.g., *or*, *once*, *eventually*) require case distinctions, which lead to the splitting of the reachable set computation into several branches. This will be part of future research.

The set of currently supported predicates is listed in Tab. I and Tab. II. To improve readability, we use an informal notation; a formal description is given in [27].

Tab. I. Supported predicates involving a single agent.

Predicate	Description
$\mathrm{OnLanelet}(A^i, l)$	Agent A^i is required to be on the specified lanelet $l \in \gamma^i$.
$\mathrm{OnCS}(A^i, cs)$ $\mathrm{BeforeCS}(A^i, cs)$ $\mathrm{BehindCS}(A^i, cs)$	Agent A^i is required to be on / before / behind a specified conflict section cs.
$\mathrm{VelocityLimit}(A^i)$	The velocity v of agent A^i is required to be within a permissible range.

Tab. II. Supported predicates involving multiple agents.

Predicate	Description
$\mathrm{BehindAgent}$ $(A^{i_1}, A^{i_2}, A^{i_3}, ..., A^{i_\mathrm{M}})$	Agent A^{i_1} is required to be behind A^{i_2}, which is required to be behind A^{i_3}, etc.
$\mathrm{SlowerAgent}$ $(A^{i_1}, A^{i_2}, A^{i_3}, ..., A^{i_\mathrm{M}})$	Agent A^{i_1} is required to be slower than A^{i_2}, which is required to be slower than A^{i_3}, etc.

A variety of traffic scenarios may be modeled by MTL formulas consisting of these operators and predicates. The MTL formulas are converted to sets of predicates $\mathcal{P}_k$ for each time step $\square_k$. The notations $\mathcal{P}_k^{\mathrm{single}}$ and $\mathcal{P}_k^{\mathrm{multi}}$ return the single- and multi-agent predicates at time step $\square_k$, respectively.

III. Methodology

Our synthesis of traffic scenarios consists of two steps. First, for each agent, its specification-compliant, forward-consistent, convex reachable sets are computed for each time step (see Sec. III-A – III-C). Second, a QP optimization synthesizes a concrete trajectory for each agent within these reachable sets (see Sec. III-D).

A. Concept for Computing the Reachable Sets

A state x_{k+1}^i is defined to be *reachable* in this work if its predecessor x_k^i is in the previous reachable set $\mathcal{R}_k^i$, the input u_k^i is in the permissible interval $\mathcal{U}^i$, and the system dynamics (2) are adhered to:

$$\mathcal{R}_{k+1}^i = \{A_\mathrm{s} x_k^i + B_\mathrm{s} u_k^i \mid x_k^i \in \mathcal{R}_k^i, u_k^i \in \mathcal{U}^i\}. \tag{3}$$

Specification-compliant sets $\tilde{\mathcal{R}}_k^i$ are subsets of the reachable sets $\mathcal{R}_k^i$ whose states that fulfill the predicates $p \in \mathcal{P}_k$ with respect to the specification-compliant reachable sets of the other agents $\tilde{\mathcal{R}}_k^{\{1,2,...,\mathrm{N}\}\setminus \mathrm{i}}$

$$\begin{aligned}\tilde{\mathcal{R}}_k^i = \{x_k^i \in \mathcal{R}_k^i \mid \forall p \in \mathcal{P}_k\ (p(x_k^1, x_k^2, ..., x_k^\mathrm{N})),\\ \forall j \in \{1,2,...,\mathrm{N}\}\ (x_k^j \in \tilde{\mathcal{R}}_k^j)\}.\end{aligned} \tag{4}$$

Forward consistency ensures that for each state in a set, there exists a dynamically feasible trajectory that leads to a state in the final set. We use a recursive, one-step forward consistency definition. This is computed backwards in time, starting at the final time step $\square_\mathrm{f}$ with [28]:

$$\begin{aligned}\overset{\frown}{R}_\mathrm{f}^i &= \tilde{\mathcal{R}}_\mathrm{f}^i,\\ \overset{\frown}{R}_k^i &= \{x_k^i \in \tilde{\mathcal{R}}_k^i \mid x_{k+1}^i = A_\mathrm{s} x_k^i + B_\mathrm{s} u_k^i,\\ &\qquad u_k^i \in \mathcal{U}^i, x_{k+1}^i \in \overset{\frown}{R}_{k+1}^i\}.\end{aligned} \tag{5}$$

By limiting the states x_k^i to specification-compliant reachable sets $\tilde{\mathcal{R}}_k^i$ in (5), the resulting forward-consistent reachable sets $\overset{\frown}{R}_k^i$ are specification-compliant. In the remainder of this paper, we refer to all types as *reachable sets* and specify the precise type by its corresponding symbol.

For an efficient computation of the reachable sets, the procedure is split into two parts: First, specification compliance and forward propagation are handled in the *forward path* (see Alg. 1). Subsequently, the *backward path* ensures forward consistency. As the initial sets are convex and all applied operations (intersection, linear transformation, Minkowski sum) preserve convexity [29], [30], all sets are convex.

Input: Initial sets $\mathcal{R}_0^{\{1,2,...,\mathrm{N}\}}$,
formal specifications $\mathcal{P}_{\{0,1,...,\mathrm{f}\}}^{\mathrm{single}}$, $\mathcal{P}_{\{0,1,...,\mathrm{f}\}}^{\mathrm{multi}}$
Output: Reachable sets $\overset{\frown}{\mathcal{R}}_{\{0,1,2,...,\mathrm{f}\}}^{\{1,2,...,\mathrm{N}\}}$

▷ Forward path
for $k \in \{0, 1, ..., \mathrm{N}\}$ **do**
  $\tilde{\mathcal{R}}_k^{\{1,2,...,\mathrm{N}\}} \leftarrow \mathcal{R}_k^{\{1,2,...,\mathrm{N}\}}$ ▷ Initialization
  for $p_j \in \mathcal{P}_k^{\mathrm{single}}$ ▷ Spec. comp. – single agent $\square^i$ **do**
    $\tilde{\mathcal{R}}_k^i \leftarrow \tilde{\mathcal{R}}_k^i \cap \mathcal{S}_{j,k}^i$
  for $p_j \in \mathcal{P}_k^{\mathrm{multi}}$ ▷ Spec. comp. – multi agent **do**
    $\tilde{\mathcal{R}}_k^{\{1,2,...,\mathrm{N}\}} \leftarrow \tilde{\mathcal{R}}_k^{\{1,2,...,\mathrm{N}\}} \cap \mathcal{S}_{p,k}^{\{1,2,...,\mathrm{N}\}}$
  for $i \in \{1, 2, ..., \mathrm{N}\}$ ▷ Forward propagation **do**
    $\mathcal{R}_{k+1}^i \leftarrow A_\mathrm{s} \mathcal{R}_k^i \oplus B_\mathrm{s} \mathcal{U}^i$ ▷ (6)

▷ Backward path
$\overset{\frown}{\mathcal{R}}_\mathrm{f}^{\{1,2,...,\mathrm{N}\}} \leftarrow \tilde{\mathcal{R}}_\mathrm{f}^{\{1,2,...,\mathrm{N}\}}$ ▷ Initialization
for $i \in \{1, 2, ..., \mathrm{N}\}$ **do**
  for $k \in \{\mathrm{f}-1, ..., 1, 0\}$ ▷ Forward consistency **do**
    $\overset{\frown}{\mathcal{R}}_k^i \leftarrow\ A_\mathrm{s}^{-1}(\overset{\frown}{\mathcal{R}}_{k+1}^i \oplus -B_\mathrm{s}\mathcal{U}^i) \cap \tilde{\mathcal{R}}_k^i$ ▷ (8)

Alg. 1. Computation of specification-compliant, forward-consistent, convex reachable sets.

B. Forward Path of Reachability Analysis

1) Forward Propagation: The forward propagation implements the one-step reachability $\mathcal{R}_k^i \to \mathcal{R}_{k+1}^i$, as defined in (3), using the Minkowski sum[2]:

$$\mathcal{R}_{k+1}^i = A_\mathrm{s} \mathcal{R}_k^i \oplus B_\mathrm{s} \mathcal{U}^i. \tag{6}$$

2) Restricting Sets to Comply with Predicate Specifications: In classic MTL, the fulfillment of a predicate p is evaluated on concrete states [26]. As the fulfillment of a

[2] $\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$

predicate can differ for various states within a set, evaluating predicates cannot be readily extended to sets. One approach to achieving specification compliance for reachable sets $\tilde{\mathcal{R}}_k^i$ is to restrict them to those sets that assuredly fulfill the predicates as defined in (4) [27].

To this end, the permissible state space intervals of the affected agents $\mathcal{S}_{j,k}^i$ are determined that fulfill a predicate $p_j \in \mathcal{P}_k$. Subsequently, the corresponding specification-compliant reachable set is updated $\tilde{\mathcal{R}}_k^i \leftarrow \tilde{\mathcal{R}}_k^i \cap \mathcal{S}_{j,k}^i$ (see Alg. 1).

For single-agent predicates, the intervals $\mathcal{S}_{j,k}^i$ are immediately accessible in our implementation due to the preprocessing (see Sec. II). E.g., for the $\mathrm{OnLanelet}(A^i, l)$ predicate, the longitudinal position interval $[\underline{s}^i, \overline{s}^i]$, for which an agent A^i intersects with the lanelet l, is read from the route information of the agent (see Sec. II-B.1).

For predicates involving multiple agents, a partition of the state space among the agents is required if their reachable sets $\mathcal{R}_k^i$ intersect, to determine the permissible intervals $\mathcal{S}_{j,k}^i$. We present a simple implementation below that ensures specification compliance using a heuristic to underapproximate the reachable sets $\tilde{\mathcal{R}}_k^i$. More sophisticated approaches are conceivable that, e.g., utilize negotiation strategies [31]. Our implementation is presented exemplarily for $\mathrm{SlowerAgent}(A^1, A^2, A^3, A^4)$. This predicate requires that A^1 is the slowest agent and A^4 the fastest one, with a nonintersecting partition of the velocity domain of the state space among all agents. Our approach consists of three steps:

1) The reachable velocity interval of each agent is determined as the projection of reachable sets $\mathcal{R}_k^i$ onto the velocity domain (see Fig. 4: outer rectangles).
2) Intersections of the velocity intervals of succeeding agents $[\underline{v}_j, \overline{v}_j]$ are determined, with j counting the partitions (see Fig. 4: interval symbols between rectangles).
3) Thresholds for the partitions of the state space among the agents are determined (see Fig. 4: horizontal lines connecting two rectangles). With M being the number of involved agents, the threshold values v_j are

$$v_j = \left(1 - \frac{j}{\mathrm{M}}\right) \underline{v}_j + \frac{j}{\mathrm{M}} \overline{v}_j. \tag{7}$$

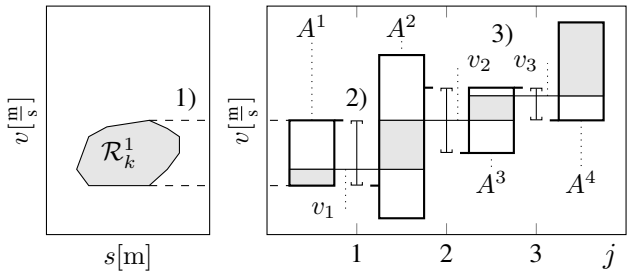

Fig. 4. Partition of velocity intervals for $\mathrm{SlowerAgent}(A^1, A^2, A^3, A^4)$ predicate.

For predicates of the type $\mathrm{BehindAgent}$, the partition procedure is similar. However, the longitudinal offset of the different route-based coordinate frames Γ^i and the length of the agents must be considered when determining the intersection of the position intervals. Predicates describing nonlinear relations, e.g., safe distance predicates [21], can be handled similarly to [27].

3) Order of Predicate Consideration: In Alg. 1, the single-agent predicates are considered prior to the multi-agent ones. Single-agent predicates are evaluated precisely, whereas considering multi-agent predicates underapproximates the left-over reachable sets $\tilde{\mathcal{R}}_k^i$. With this order of predicate consideration, we reduce the approximation effect.

C. Backward Path of the Reachability Analysis

We remove non forward-consistent states in the backward path. This is achieved by intersecting the specification-compliant reachable set $\tilde{\mathcal{R}}_k^i$ with $\overleftarrow{\mathcal{R}}_k^i$ according to (5). This procedure is carried out iterating backward, starting with $k = \mathrm{f} - 1$ (see Alg. 1). Our implementation uses the Minkowski sum to compute the backward propagation [32]:

$$\mathcal{R}_k^i = A_\mathrm{s}^{-1}(\overleftarrow{\mathcal{R}}_{k+1}^i \oplus - B_\mathrm{s}\mathcal{U}^i) \cap \tilde{\mathcal{R}}_k^i. \tag{8}$$

The inverse of A_s always exists for discrete-time linear systems [33, Sec. 7.2]. A more detailed description of forward consistency computation is given in [28]. The resulting reachable sets $\overleftarrow{\mathcal{R}}_k^i$ are the input to the QP optimization trajectory synthesis.

D. Trajectory Synthesis

To derive the trajectory of each agent, a QP optimization is solved for each agent A^i individually. The objective function is the sum of squared accelerations (9a). As constraints, the system dynamics of the agent (2) must be considered (9b). The search space of the optimization is given by the reachable sets $\overleftarrow{\mathcal{R}}_k$ (9c).

$$\min_{u_k^i} \; J = \sum_{k=0}^{\mathrm{f}-1} {u_k^i}^2 \tag{9a}$$

$$\text{s.t.} \quad x_{k+1}^i = A_\mathrm{s} x_k^i + B_\mathrm{s} u_k^i \;\; \forall k \in \{0, 1, ..., \mathrm{f}-1\}, \tag{9b}$$

$$x_k^i \in \overleftarrow{\mathcal{R}}_k^i \qquad \forall k \in \{0, 1, ..., \mathrm{f}\}. \tag{9c}$$

For modeling the convex reachable sets $\overleftarrow{\mathcal{R}}_k^i$ as constraints (9c), the vertex representation used in previous computations is converted to the half-space representation [34]. As the reachable sets $\overleftarrow{\mathcal{R}}_k^i$ consider dynamical limitations and formal specifications, the optimizations for the different agents are simultaneously executable, reducing the computation time. With the trajectory synthesis being defined, the pipeline for synthesizing traffic scenarios for formal specifications is complete. Crucial for the success of our approach is its performance, which is assessed in the following section.

IV. Numerical Results

Two numerical examples are presented to assess the results of the reachability-based scenario synthesis. Both examples are based on [18]. The first one describes a lane change and merge maneuver, and the second one specifies the order in which traffic participants pass the conflict section on a T-junction. The solution quality and computational

performance are compared to the MIQP approach from [18], which is well-suited as a reference due to its similar definitions and functionality. For both examples, the lanelet networks are available on our website[3] as *ZAM_Zip*, and *ZAM_Tjunction*. All agents have the same limits for acceleration $a = [-6, 3]\,\mathrm{m\,s^{-2}}$, and velocity $v = [0, 30]\,\mathrm{m\,s^{-1}}$. The time increment in both examples is $\Delta t = 0.25\,\mathrm{s}$.

A. Merge Scenario

In the merge scenario, two adjacent and equally directed lanes merge. This occurs, e.g., with lane narrowing in urban traffic or with road works on highways. Four agents appear in the traffic scenario (see Fig. 5). Initially, there are two vehicles in each of the adjacent lanes. The rear vehicle in the left lane A^2 then changes lanes and moves to the front of the vehicles in the right lane. Subsequently, the vehicle remaining in the left lane A^1 is to merge with the right lane in front of the last vehicle A^4. The formal description is provided in Tab. III and the time horizon is $\mathrm{f} = 40$ time steps.

Fig. 5. Lanelet network, reference paths, and initial positions of the agents.

Tab. III. Predicate sets $\mathcal{P}_k$ of merge scenario.

Time step k	Predicates p
0...40	VelocityLimit($\{A^1, A^2, A^3, A^4\}$), BehindAgent(A^4, A^3)
0	BehindAgent(A^2, A^1)
15...30	BehindAgent(A^1, A^2)
30...40	BehindAgent(A^4, A^1, A^3, A^2)
40	OnLanelet($A^1, 5$), SlowerAgent(A^2, A^1)

One can observe that agent A^1 must temporarily reduce its velocity substantially so that the correct sequence of agents can be achieved (see Fig. 6). Agent A^2 initially has a high velocity before it has to reduce its velocity to comply with the SlowerAgent(A^2, A^1) predicate. As a consequence, the reachable set $\overset{\frown}{\mathcal{R}}{}^2_{30}$ is considerably restricted, since higher velocities at this time step would lead to exceeding the allocated velocity range at the time step $k = 40$ under the given acceleration limits. In contrast, agent A^3 is not affected by such a predicate and its reachable set $\overset{\frown}{\mathcal{R}}{}^3_{40}$ covers the whole permissible velocity range.

After the reachable set computation, the QP optimization for synthesizing the trajectories is conducted. By definition, the resulting trajectories lie within the previously computed sets in the state space representation (see solid lines in Fig. 6). The objective value (9a) of the MIQP optimization is $J_{\mathrm{MIQP}} = 150.7\,\mathrm{m^2 s^{-4}}$, whereas the reachability approach incurs an objective value of $J_{\mathrm{Reach+QP}} = 264.5\,\mathrm{m^2 s^{-4}}$.

[3]https://commonroad.in.tum.de/scenarios

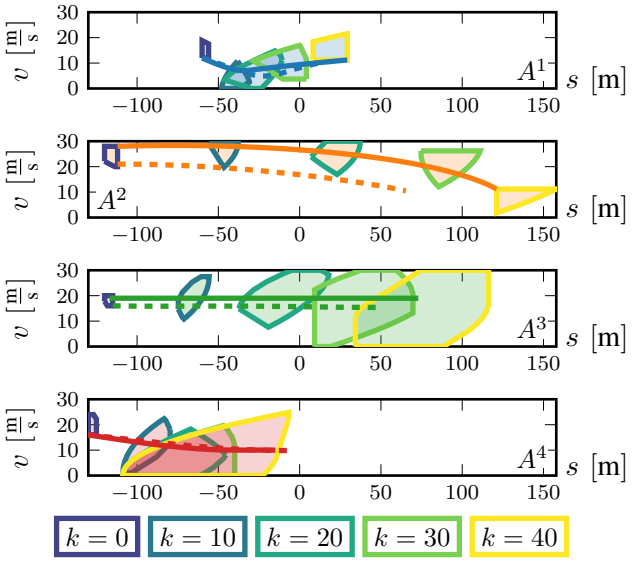

Fig. 6. Reachable sets $\overset{\frown}{\mathcal{R}}{}^i_k$ over time $t = \{0, 2.5, 5, 7.5, 10\}\,\mathrm{s}$ (outline color). Routes intersect at merge m (see Fig. 5) with $s = 0\,\mathrm{m}$. Solid trajectories: Reach + QP; Dashed trajecories: MIQP.

Apart from that, the trajectories are similar in the sense that they both comply with the formal specifications. The different objective values result from the ability of the MIQP optimization to adjust the threshold value of the multi-agent predicates as part of its optimization. In contrast, the reachability-based approach chooses a heuristic value which is not necessarily optimal (see Sec. III-B.2). With more sophisticated negotiation strategies for considering the predicates [31], a solution closer to the optimal one can be expected. The computation performance aspect will be assessed in Sec. IV-C. Before, the second numerical example is presented.

B. T-Junction Scenario

The next example considers a T-junction where the routes of six agents interfere with each other (see Fig. 7). Therefore, a safe order in which the agents pass the conflict section cs must be specified. This is ensured by enforcing that all agents except one are either BeforeCS or BehindCS (see Tab. IV).

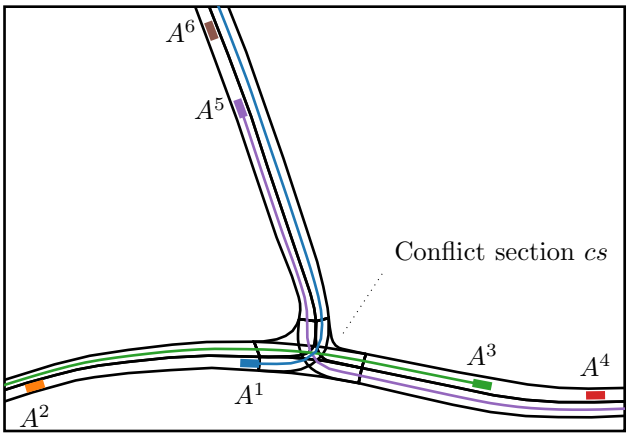

Fig. 7. Lanelet network, reference paths, and initial positions of agents. Agents that are initialized in the same lanelet follow the same routes. Only the routes of the leading vehicles are depicted.

Tab. IV. Predicate sets $\mathcal{P}_k$ of T-junction scenario.

Time step k	Predicates p	
0...48	VelocityLimit($\{A^1, A^2, A^3, A^4, A^5, A^6\}$), BehindAgent($\{(A^2, A^1), (A^4, A^3), (A^6, A^5)\}$)	
	BeforeCS($\square, cs$)	BehindCS($\square, cs$)
0	$\{A^1, A^2, A^3, A^4, A^5, A^6\}$	$\{\}$
8	$\{A^2, A^3, A^4, A^5, A^6\}$	$\{A^1\}$
16	$\{A^2, A^4, A^5, A^6\}$	$\{A^1, A^3\}$
24	$\{A^2, A^4, A^6\}$	$\{A^1, A^3, A^5\}$
32	$\{A^4, A^6\}$	$\{A^1, A^2, A^3, A^5\}$
40	$\{A^6\}$	$\{A^1, A^2, A^3, A^4, A^5\}$
48	$\{\}$	$\{A^1, A^2, A^3, A^4, A^5, A^6\}$

Once again, the MIQP achieves a substantially lower objective value of $J_{\text{MIQP}} = 138\,\text{m}^2\text{s}^{-4}$ compared to $J_{\text{Reach+QP}} = 503\,\text{m}^2\text{s}^{-4}$. However, the trajectories of both approaches have a similar shape and are smooth (see Fig. 8).

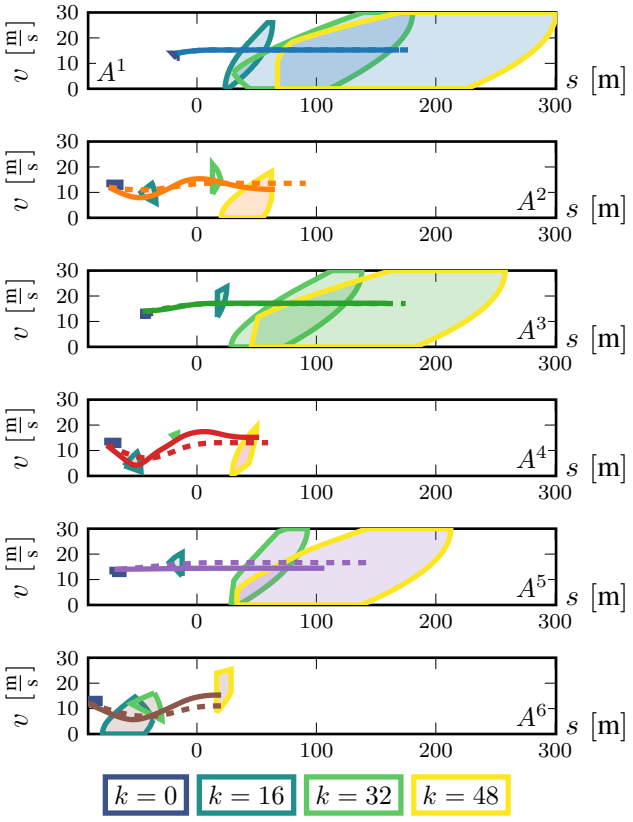

Fig. 8. Reachable sets $\widehat{\mathcal{R}}_k^i$ over time $t = \{0, 4, 8, 12\}$ s (outline color). Routes intersect at conflict section cs with $s = 0\,\text{m}$ (see Fig. 7). Solid trajectories: Reach + QP; Dashed trajecories: MIQP.

C. Performance Comparison to MIQP Approach

The source code for both implementations, MIQP and reachability-based, is written in Julia. Moreover, a package[4] provides the interface to the Gurobi optimizer. All timings are conducted with an Intel i7-12700H processor.

[4]https://github.com/jump-dev/Gurobi.jl

Besides a general performance comparison, analyzing the scalability of both approaches with increasing traffic scenario complexity is of particular interest. Varying the complexity is achieved by terminating the T-junction scenario after differing durations, meaning that the number of vehicles that pass the intersection differs (see Tab. V).

Tab. V. Computation times τ and objective values J with MIQP and reachability-based approach for different scenario complexities.

Number of agents passing the conflict section	1	2	3	4	5	6
Termination at k	8	16	24	32	40	48
Scenario duration [s]	2	4	6	8	10	12
Binary variables	403	615	867	1159	1491	1863
J_{MIQP} [m^2s^{-4}]	0	0	20.7	54.4	119	138
$J_{\text{Reach+QP}}$ [m^2s^{-4}]	4.11	16.7	38.3	128	365	503
τ_{MIQP} [10^3 ms]	.062	.117	.266	1.07	2.122	3.09
τ_{Reach} [ms]	.116	.241	.414	.623	.860	1.08
$\tau_{\text{Reach+QP}}$ [ms]	1.51	3.02	4.94	6.98	9.49	11.7
Speed-up $= \frac{\tau_{\text{MIQP}}}{\tau_{\text{Reach+QP}}}$	40.9	38.6	53.9	153	224	264

The computation times of the MIQP implementation are similar to those in the original paper [18]. The computation times of the MIQP optimization increase exponentially with the number of binary variables [19]. Moreover, depending on how well the solving strategies of Gurobi work on the given problem, the solution time might vary [35]. This leads to nondeterministic execution times.

The computation time of the reachability approach increases approximately linearly with the scenario duration. Compared to the exponential increase for the MIQP approach, this leads to substantially faster computation times, especially for scenarios with long durations and many agents. Moreover, less than $10\,\%$ of the computation time is spent with the reachable set $\widehat{\mathcal{R}}_k^i$ computation. Instead, most time is used for solving the QP optimization for synthesizing the trajectories. The QP optimization for the different agents is carried out in parallel on different cores. This reduces the overall computation time by a factor of about three. Within the reachable set computation, the computation time is divided equally among the application of the predicates, the forward propagation, the backward propagation, and the intersection of the sets. Thus, the forward and backward paths take nearly the same amount of time.

V. Conclusions

A novel approach for synthesizing traffic scenarios from formal specifications is presented. First, the reachable sets of the agents that comply with the formal specifications are computed. In addition, using backward propagation, these sets are limited to those states that are forward-consistent. Subsequently, within these sets, trajectories for all agents are synthesized using QP optimization.

The major limitation of this approach is that the optimality of the solution cannot be guaranteed. In fact, for the investigated examples, the objective value of the reachability-based

trajectory synthesis increases considerably compared to the optimal solution. This could be mitigated by an improved partition of the reachable sets when applying predicates among interfering agents, e.g., using negotiation strategies [31], [36].

In contrast, the reachability-based approach offers substantial advantages over other state-of-the-art trajectory synthesis techniques. First, the computation times scale linearly with increasing scenario duration and complexity. In addition, the duration of the time-consuming QP trajectory synthesis can be kept constant for increasing numbers of agents by solving the individual optimization problems for each agent on multiple cores in parallel. The combination of both aspects makes our approach highly suitable for complex traffic scenarios with many agents.

REFERENCES

[1] M. A. Schreurs and S. D. Steuwer, "Autonomous driving - political, legal, social, and sustainability dimensions," in *Autonomous Driving - Technical, Legal and Social Aspects*. Springer Berlin Heidelberg, 2015, ch. 8, pp. 151–173.

[2] A. Gräter, M. Harrer, M. Rosenquist, and E. Steiger, *Connected, Cooperative and Automated Mobility Roadmap*, 10th ed. European Road Transport Reserach Advisory Council, 2022.

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

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

[5] ISO Central Secretary, "Road vehicles - safety of the intended functionality," International Organization for Standardization, Geneva, CH, Standard ISO 21448:2022(E), 2022.

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

[7] W. Ding, C. Xu, M. Arief, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical driving scenario generation—a methodological perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 6971–6988, 2023.

[8] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles," in *IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2383–2390.

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

[10] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *IEEE Intelligent Vehicles Symposium*, 2018, pp. 1813–1820.

[11] J. S. Eo, H. R. Choi, R. Gao, S.-y. Lee, and W. E. Wong, "Case study of requirements-based test case generation on an automotive domain," in *IEEE International Conference on Software Quality, Reliability and Security*, 2015, pp. 210–215.

[12] Y. Akagi, R. Kato, S. Kitajima, J. Antona-Makoshi, and N. Uchida, "A risk-index based sampling method to generate scenarios for the evaluation of automated driving vehicle safety," in *IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 667–672.

[13] F. Klück, Y. Li, M. Nica, J. Tao, and F. Wotawa, "Using ontologies for test suites generation for automated and autonomous driving functions," in *IEEE International Symposium on Software Reliability Engineering Workshops*, 2018, pp. 118–123.

[14] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, "From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment," in *IEEE Intelligent Vehicles Symposium*, 2019, pp. 2383–2390.

[15] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, 2020.

[16] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *IEEE Conference on Decision and Control*, 2008, pp. 2117–2122.

[17] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 5319–5325.

[18] M. Klischat and M. Althoff, "Synthesizing traffic scenarios from formal specifications for testing automated vehicles," in *IEEE Intelligent Vehicles Symposium*, 2020, pp. 2065–2072.

[19] Y. Shimai, J. Tani, H. Noguchi, H. Kawaguchi, and M. Yoshimoto, "FPGA implementation of mixed integer quadratic programming solver for mobile robot control," in *International Conference on Field-Programmable Technology*, 2009, pp. 447–450.

[20] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

[21] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[22] J. Nocedal and S. J. Wright, "Theory of constrained optimization," in *Numerical Optimization*, 2nd ed. Springer New York, 2006, pp. 304–354.

[23] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[24] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 1–35, 2022.

[25] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[26] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[27] E. I. Liu and M. Althoff, "Computing specification-compliant reachable sets for motion planning of automated vehicles," in *IEEE Intelligent Vehicles Symposium*, 2021, pp. 1037–1044.

[28] M. Klischat and M. Althoff, "Falsifying motion plans of autonomous vehicles with abstractly specified traffic scenarios," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1717–1730, 2023.

[29] D. D. Bremner, "On the complexity of vertex and facet enumeration for convex polytopes," Ph.D. dissertation, CAN, 1998.

[30] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling, "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *International Conference on Hybrid Systems: Computation and Control*. Association for Computing Machinery, 2018, pp. 41–50.

[31] E. Irani Liu and M. Althoff, "Specification-compliant driving corridors for motion planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, pp. 1–17, 2023, early access.

[32] B. Schürmann, M. Klischat, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 5698–5713, 2022.

[33] E. B. Saff and A. D. Snider, *Fundamentals of matrix analysis with applications set*. Nashville, TN: John Wiley & Sons, 2016.

[34] B. Grünbaum, "Polytopes," in *Convex Polytopes*. Springer New York, 2003, pp. 35–60.

[35] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter, "MIPLIB 2010," *Mathematical Programming Computation*, vol. 3, no. 2, pp. 103–163, 2011.

[36] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 444–451.