A Knowledge Architecture Layer for Map Data in Autonomous Vehicles

Haonan Qiu*†, Adel Ayara*, Birte Glimm†
*BMW Car IT GmbH, Ulm, Germany, Email: firstname.lastname@bmw.de
†Institute of Artificial Intelligence, University of Ulm, Germany, Email: firstname.lastname@uni-ulm.dm

***Abstract*— Autonomous Driving (AD) systems use digital maps as a virtual sensor to perceive the environment around the car. As the field of digital maps continues to evolve, existing solutions face new challenges such as integration ability for new map formats (e.g., High Definition maps), supporting onboard and offboard deployment and providing a generic interface to access the road environmental knowledge. In this paper, we propose a knowledge architecture layer for environmental modeling and distinguish between low-level ontologies based on various map data formats and a high-level ontology for representing a generic road environment. The adequacy of the modeling is validated over two use cases: lane change notification and logical inconsistency detection. The performance is measured using real map data and it shows encouraging results for future development within onboard and offboard systems.**

I. Introduction

Autonomous Driving (AD) systems use digital maps as a virtual sensor to anticipate the road ahead, understand and navigate the vehicles' environment and make decisions [1]. Current implementations of map data processing are tightly coupled to the underlying raw data formats, which results in poor extensibility. Applications and protocols using standard definition (SD) maps such as the Advanced Driver Assistance System Interface Specification (ADASIS) v2 cannot easily be extended to handle High Definition (HD) maps [2]. Besides the different requirements for HD and SD maps, it is also not clear how to implement a solution in an interoperable way using data from different map providers, e.g., TOMTOM, HERE, or Google. Moreover, HD maps have to be deployed not only onboard but also on the backend to ensure continuously updated map data. These requirements are the new challenges for map data and related applications and call for a solution that provides i) a semantic representation of map data to break the dependency of raw data; ii) a solution that can be deployed onboard and offboard; iii) a generic interface to access the road environmental knowledge.

To address these challenges, we propose an ontology-based approach with rules to empower, in particular, autonomous vehicles with environmental cognition using map data. Guarino et al. [3] provide a summarized definition of ontologies as a formal, explicit specification of a shared conceptualization. The use of ontologies to describe semantic, temporal, and spatial aspects of the environment for robotics is well studied [4]. Ontologies structure the robotic environment knowledge by describing the individual elements and relationships in the domain using unary and binary predicates. An ontology constructed from different levels of abstraction is able to integrate different ontological approaches in a unified system, and it is commonly practised in Medical and Biomedical domains [5] and Context-Aware Systems[6], [7]. Inspired by multi-levels of ontological abstraction, we describe our approach for processing map data in autonomous vehicles. The contributions of this paper can be summarized as follows:

– We propose a knowledge architecture with two levels of abstraction to solve the map data integration problem.
– We provide a generic high-level ontology to represent road environmental knowledge.
– We show the usage of datalog rules for knowledge transfer and maneuver decision making.

Our prototype called SmartMapApp uses the developed OWL 2 RL [8] ontologies and datalog rules [9]. The evaluation employs two use cases in a highway scenario: lane change notification and inconsistency detection and we further present performance results for three reasoning tasks.

The remainder of this paper is organized as follows: Section II presents related works. In Section III, we discuss the ontology-based SmartMapApp prototype, which is evaluated in Section IV. Section V concludes the paper.

II. Related Work

An environmental model is needed for AD systems to be aware of the situation around a vehicle. Ontology-based road environment modeling has gained growing interest in the Intelligent Transportation Systems community [10]. The developed road models provide different features with respect to specific application requirements.

Hummel et al. developed an ontology for road and intersection understanding [11], which provides background knowledge for vision sensors and digital maps. Ontologies have further been used for representing road intersections for autonomous vehicles [12]–[14]. An ontology-based driving decision making system was built within safety Advanced Driver Assistance Systems (ADAS) on uncontrolled intersections. The constructed ontologies contain three main ontologies: the map, the control, and the core ontology [15]. These approaches allowed for modeling the road environment as well as reasoning over road situations. Armand et al. [16] proposed a framework for ontology-based situation understanding, where digital maps are used to provide prior knowledge about the environment. Combined with the application of ontologies, the system can provide meaning to the relationship between sensor perceived road objects.

The mentioned approaches represent the road environment using ontologies to provide both an explicit and an implicit

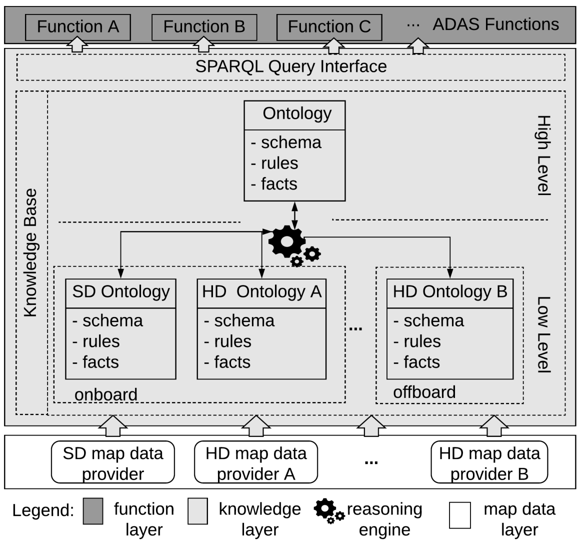

Fig. 1. System architecture with two levels of ontologies

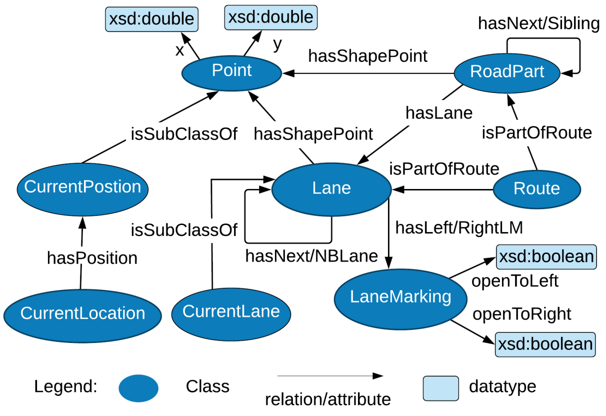

Fig. 2. Overview of the high-level ontology, where hasNBLane abbreviates hasNeighboringLane and hasLeftLM abbreviates hasLeftLaneMarking

specification of context information for vehicles. However, their main limitation is the inability to take the integration of different map data formats into account, as well as the dynamic aspect of map data (e.g. car positions). Suryawanshi et al. [17] also proposed a separation of the underlying data structures and the represented knowledge in their work for representing SD map data in vehicles, but their work is neither generalized to different map formats nor is the spatial topology of lanes considered.

III. Ontology-Based SmartMap

Figure 1 shows the proposed knowledge architecture. We next discuss the introduced components and then describe of the designed two-level ontologies and rules in more detail.

A. Architecture

The **map data layer** comprises the external SD and HD map data sources and populates the low-level ontologies that represent the different map data formats.

The **knowledge layer** consists of two levels of ontologies. Each ontology contains its own schema, facts, and rules. This makes the architecture flexible with respect to the addition of diverse types of map data formats. The low-level ontologies represent the raw map data. Each ontology in this level describes a specific map data format and related knowledge extraction tasks. A single high-level ontology is used to represent the generic concepts and relations of maps. This ontology is used at the reasoning level for answering diverse queries for ADAS functions. Entities in the high-level ontology are produced by rules using facts in a low-level ontology. A SPARQL query interface provides a generic and unified access interface to retrieve knowledge and the queries can either be predefined or composed on-the-fly.

The **reasoning engine** takes the schema and rules as input and deals with three types of tasks: i) processing the low-level ontologies, ii) populating the high-level ontology and iii) generating dynamic vehicle related knowledge. Rules also deal with task i) and ii) and are defined in each low-level ontology using concepts and relationships related to the corresponding map formats. Rules for task iii) are defined based on concepts and relationships representing the road environment on a high level.

The **function layer** consists of various ADAS functions such as lane change assistance and turning assistant, which can easily be build on top of the knowledge layer through the SPARQL query interface.

B. High-Level Ontology

Figure 2 gives an overview of the high-level map ontology. Lane is the main concept as it provides the basic building block for describing the road environment. Each Lane is characterized by a set of Points with coordinates (x, y) in the World Geodetic System (WGS) 84 format. Each Lane has a left LaneMarking and a right LaneMarking. The important attributes of LaneMarking are openToLeft and openToRight, which are used to infer whether the vehicle is allowed to change Lane(s). RoadPart(s) are used to describe the topology of roads, and each RoadPart has its own Lane(s). CurrentLocation is described by CurrentPosition. CurrentPosition and CurrentLane are dynamic concepts related to periodically updated vehicle positions. A Route can be constituted of Lane(s), RoadPart(s) or both.

Since vehicles travel longitudinally in a single lane and move laterally when changing lanes, we classify the lane relations into two categories, namely lateral relations and longitudinal relations. With these two types of relations, we can represent a graph-based environmental model using lanes. The lateral relations of lanes describe the possible maneuvers between lanes in lateral (left/right) direction. These relations are further generalized as a “neighboring” relation. We model hasLeftLane and hasRightLane as sub-properties of hasNeighboringLane. Furthermore, it is not only necessary to know about the existence of neighboring lanes, but also whether crossing to them is allowed. This information is usually encoded in lane markings. The longitudinal relations of lanes describe the reachability of lanes in longitudinal

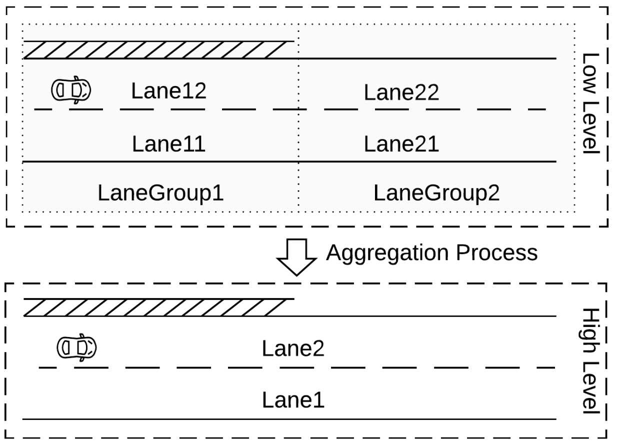

Fig. 3. Aggregation of low level lanes into high-level lanes

direction within continuous lanes. We further divide this type of relation into successor and predecessor relations. As for the lateral relations, we model hasPredecessorLane and hasSuccessorLane as sub-properties of hasNextLane.

C. Low-Level Ontologies

The low-level ontologies represent different map data formats. For example, HD map data in the NDS format is organized into building blocks and the relations between the concepts LaneGroup, Lane, and LaneBoundary represent the lane-level topology. The concept Feature and the relations of its entities represents the road-level topology. The SD map data based on the ADASIS v2 protocol, on the other hand, represents roads as collections of nodes (road segments).

Even though the low-level ontologies do not necessarily share common concepts, the entities of the ontologies can be transferred to the common high-level ontology resulting in a unified knowledge base. For example, the entities of the concept Feature from the ontology based on NDS data and the entities of the concept Node in ontology based on the ADASIS V2 protocol are represented by entities of the concept RoadPart in the high-level ontology.

D. Reasoning

Reasoning plays a central role in our architecture and is used to transfer entities from low-level to high-level, thereby aggregating lanes and lane markings, and to provide the needed knowledge for the customer functions to take decisions, e.g., when to notify the driver to change lanes in order to exit a highway.

IV. Evaluation

In this section, we analyse the effects of the above mentioned lanes and lane markings aggregating and then we evaluate the adequacy of the ontologies for the use cases of lane change notifications and inconsistency detection. Finally, we present the experimental results for the tasks of map initialization and current lane identification.

A. Lane and Lane Marking Aggregation

Figure 3 shows an example of lane aggregation based on the NDS specification. LaneGroup1 and LaneGroup2 both contain two lanes and the same side of lanes in each lane group is contiguous and longitudinally connected. For instance, Lane22 is the contiguous successor lane of Lane12 and they are longitudinally connected. The difference of LaneGroup1 and LaneGroup2 is that Lane12 has a guardrail as lane boundary, while Lane22 does not. In this case, human beings perceive Lane12 and Lane22 as one lane regardless of the presence of guardrails. We followed the human perception for our modeling by applying the aggregation rules, which also reduces the amount of data. We overall use three aggregation rules and Listing 1 shows the rule (in Datalog syntax [18]), which aggregates the low-level entities Lane12 and Lane22 into the high-level entity Lane2 and Lane11 and Lane21 into Lane2.

Figure 4 shows the result of the aggregation rules over map data covering $63.75\,\mathrm{km}^2$, considering motorway and urban scenarios. The number of high-level Lane entities decreases after the aggregation process in both scenarios. For motorways, the reduction rate of the Lane and LaneBoundary entities can reach 50% which helps to decrease the amount of stored data, and increases processing efficiency. In the urban scenario, the ratio is less visible due to the nature of topology and geometry of the roads inside the city.

B. Use Case: Lane Change Notification

A lane change is defined as a driving maneuver that shifts a vehicle from one lane to another where both lanes have the same travel direction. Consider the following scenario in which a vehicle is driving on a highway. The AD system realizes that the vehicle will need to exit the highway in 3 km (foresight parameter) based on the route. However, the exit lane is not longitudinally reachable by the lane which the vehicle is on. Then, the AD system generates a lane change notification in order to prepare the driver to exit the highway.

Figure 5 illustrates a simple example of the above described scenario: Lane1–Lane5 are high-level Lane entities. The vehicle is driving in Lane2. As Lane4 is the successor lane of Lane2, they are in the hasNextLane relationship. Lane1 has two successor lanes: Lane4 and Lane3, hence, Lane1 is in hasNextLane relation with Lane4 and Lane3.

According to the given route, the vehicle has to reach Lane3 from Lane2. The distance from the current position of the vehicle to the starting point of Lane5, which is the neighbouring lane of Lane3, is 3 km (foresight parameter). The lane marking between Lane2 and Lane1 is a dashed line, interpreted as being crossable. This indicates that the vehicle is allowed to perform a lane change and the AD system sends a lane change notification: *move one lane to the right.*

The key ontology components for aggregation are: hasNeighbouringLane, hasNextLane, and the combination of both. The aggregation uses thirteen rules and a SPARQL query (see Listing 2), which checks if a lane change is possible and needed: i) The current lane cannot reach a part of the route via the hasNextLane relation, we call that

```
Lane[?highLane]:-
LaneGroup[?lg1], direction[?lg1, ?dir], numLane[?lg1, ?numLane], hasLane[?lg1, ?lowlane1],
  hasLaneConnElemType[?lowlane1, ?laneType], index[?lowlane1, ?index], hasLaneBoundary[?lowlane1, ?lb1],
  numParallelElem[?lb1, ?numPara], contains[?lb1, ?paraElem1],  numSequentialElem[?paraElem1, ?numSeq],
  consistsOf[?paraElem1, ?seqElem1], openToCurbSide[?seqElem1, ?openToCurbSide],
  openToMiddleSide[?seqElem1, ?openToMiddleSide],
LaneGroup[?lg2], direction[?lg2, ?dir], numLane[?lg2, ?numLane], hasLane[?lg2, ?lowlane2],
  hasLaneConnElemType[?lowlane2, ?laneType], index[?lowlane2, ?index], hasLaneBoundary[?lowlane2, ?lb2],
  numParallelElem[?lb2, ?numPara], contains[?lb2, ?paraElem2], numSequentialElem[?paraElem2, ?numSeq],
  consistsOf[?paraElem2, ?seqElem2], openToCurbSide[?seqElem2, ?openToCurbSide],
  openToMiddleSide[?seqElem2, ?openToMiddleSide],
BIND(SKOLEM("highLane", ?dir, ?numLane, ?laneType, ?index) AS ?z),
BIND(IRI(CONCAT("http://www.bmw-carit.de/SmartMapApp/High#", str(?z))) AS ?highLane).
```

Listing 1. The rule used to aggregate the low-level lanes into high-level lanes

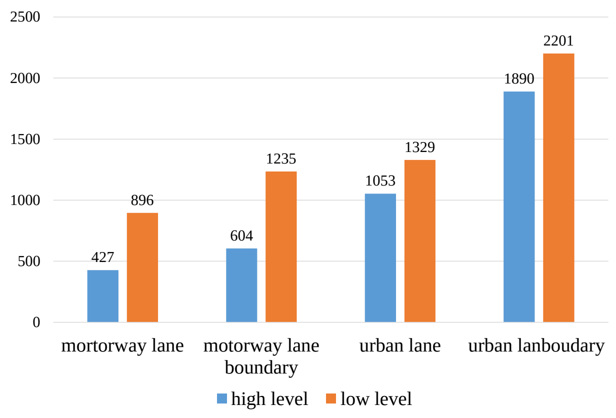

Fig. 4. Aggregation over lanes and lane boundaries in motorway and urban scenarios

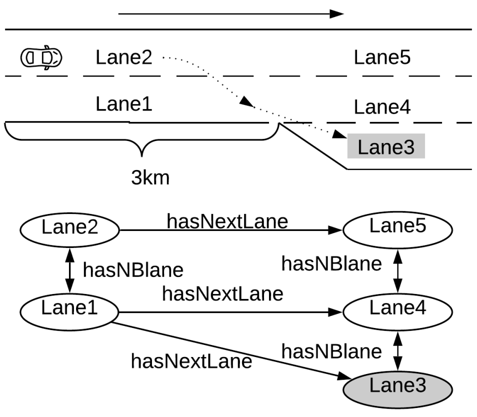

Fig. 5. A simple example of lane change scenario with the relations of lanes, hasNextLane and hasNBlane (hasNeighbouringLane)

part of the route the target lane; ii) the current lane is in hasNeighbouringLane relation with another lane, and it is allowed to move to this lane; iii) the neighbouring lane of the current lane is in a hasNextLane relation with the target lane; iv) the distance from the current position of the vehicle to the starting point of the neighbouring lane of the target lane is smaller than or equal to the foresight parameter.

One of the advantages of using rules is the ability to provide explanations. The generated explanation for the given example is: i) Lane3 is part of the route, but it is not reachable from Lane2 via the hasNextLane relation; ii) Lane2 and Lane1 are in hasNeighbouringLane relation and it is allowed to cross from Lane2 to Lane1; iii) the distance from the current position to the lane change point is 3 km, which is equal to the foresight parameter.

```
SELECT ?rd ?inbtwl1 ?ibLength ?nnbl1
       ?targetLane ?ibtwtl ?d
WHERE {
  ?cl :remainingDistance ?rd.
  ?currlane :hasNotReachableRS ?targetLane ;
            :hasNBLane ?nnbl1 .
  ?nnbl1 :hasNextLane ?targetLane, ?inbtwl1 .
  ?inbtwl1 :hasNextLane  ?targetLane ;
           :length ?ibLength .
  :foresightLaneChangeDis :hasValue ?value
  { SELECT ?currlane (SUM(?length) AS ?ibtwtl)
    WHERE {
     ?currlane :hasNotReachableRS ?highLane ;
               :hasNBLane ?nextNBLane .
     ?nextNBLane :hasNextLane ?highLane ,
                                  ?inbetweenLane .
     ?inbetweenLane  :hasNextLane  ?highLane ;
                     :length ?length .
    } GROUP BY  ?currlane
  } BIND((?rd +?ibtwtl) AS ?d)
    FILTER(?d <= ?value)
}
```

Listing 2. SPARQL query to check if a lane change is possible and needed; hasNBLane stands for hasNeighbouringLane and hasNotReachableRS stands for hasNotReachableRouteSegment. Prefixes are omitted.

C. Use Case: Logical Inconsistency Detection

Fusing the data from vehicle sensors and the map can help to form a consistent environmental view of the surroundings of the vehicle. The AD system reasons on the generated environmental model to decide which actions are appropriate for specific driving situations. This use case shows that the proposed high-level ontology enables us to detect logical inconsistencies by reasoning on the environmental model based on the vehicle sensor data and the map. We illustrate this by a false emergency break scenario (see Fig. 6), which might result from mislocalization of the car's position due to, for example, noisy or non-available sensor data.

As shown in Fig. 6, the vehicle E is driving in Lane2 following vehicle C2, and vehicle C1 is driving in Lane1. Due to the non-availability of some sensor data, E (in grey) is mislocalized in Lane1 instead of Lane2. The object tracking system (OTS) of E provides a list of positions of tracked objects (C1 and C2) around E. The provided positions are relative to the position of E as E is the origin of the vehicle coordinate system. Because E is mislocalized in Lane1 and the calculated distance of E to C1 is less than the distance threshold of the autonomous emergency braking system (e.g., 80 m), the braking system of E would stop the vehicle immediately to avoid a collision with C1 "in front", although, in reality, E is driving in Lane2 with a safe distance to C2.

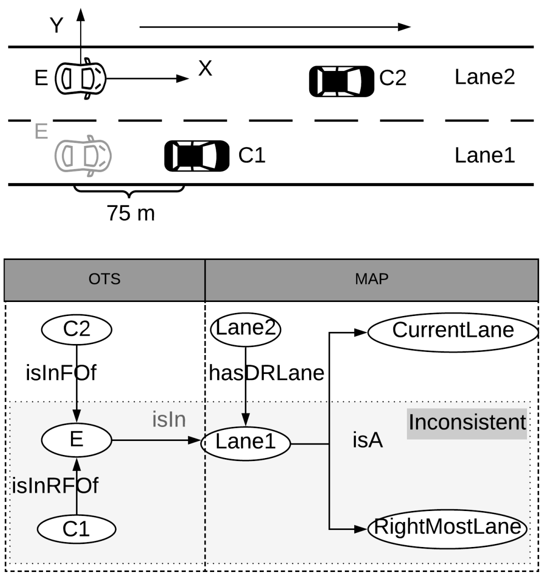

Fig. 6. Illustration of the environmental model of the ego vehicle based on the raw data provided by the Object Tracking System (OTS). The main relations of this model are isInFOf (isInFrontOf), isInRFOf (isInRightFrontOf), isIn, hasDRLane (hasDirectRightLane) and isA

This false emergency break could be avoided by creating an ontology for sensor data fused with the map data as shown in Fig. 6. The isInRightFrontOf, isInFrontOf and isIn relations are derived from sensor data and represent, respectively, that: i) C1 is in the right front lane of E; ii) C2 is in front of E on the same lane; iii) E is localized in Lane1. According to the map, the lane (Lane1) where E is localized is the rightmost lane. This contradicts the derived knowledge of isInRightFrontOf, which means that a car is in the right front lane of E, because it is impossible to have a lane at the right-hand side of the rightmost lane. Listing 3 shows the rule classifying this inconsistent knowledge as a fault.

D. Performance

For evaluating the performance of the proposed framework, we implemented SmartMapApp. The prototype currently uses two types of low-level map ontologies: i) HD map data based on the NDS standard; ii) SD map data based on the ADASIS protocol [17]. We use 14 rules for processing the low-level ontologies and 27 rules for computing inferences over the high-level ontology.

```
Fault[?fault] :-
Car[?car1], isIn[?lane], RightMostLane[?lane],
Car[?car2], isInRightFrontOf[?car2, ?car1],
BIND(SKOLEM("fault", ?car1, ?car2, ?lane) AS ?z),
BIND(IRI(CONCAT("http://www.bmw-carit.de/
    SmartMapApp/High#", str(?z))) AS ?fault).
```

Listing 3. The rule for fault detection

TABLE I
NUMBER OF TRIPLES (#TRIPLES), LOW-LEVEL LANES (#LANES), AND LANE BOUNDARIES (#LB) IN THE DATA SET AND THE COMPUTATION TIMES IN MS FOR MAP INITIALIZATION (INIT), FIRST CURRENT LANE (FIRSTPOS), AND OTHER CURRENT LANE (OTHERPOS) COMPUTATION

Data set	#Triple	#Lanes	#LB	init	FirstPos	OtherPos
1	23,647	187	374	1,558	78	19
8	104,708	897	1,794	5,356	158	35
16	429,702	3,156	6,312	49,000	4,811	59

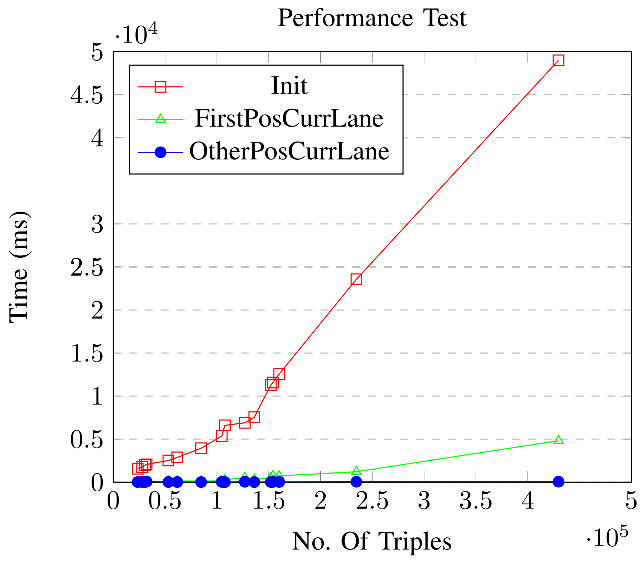

Fig. 7. Computation time for map initialization, current lane identification for the first position and the remaining positions in a trace

The used map data covers $63.75\,\mathrm{km}^2$ and is split into 16 data sets for testing. Table I shows statistics of the smallest (#1), the medium (#8) and the largest (#16) data set. The set of positions we used for simulation are represented by WGS84 coordinates stored in JSON files. For each data set, we used a trace of 10 positions to evaluate the application. SmartMapApp uses RDFox 1.6.0 [18] with the provided Java APIs. The index strategy of the datastore is set to "par-complex-nn". Six threads are allocated for importing the data and reasoning. The evaluation was performed on a 64-bit Ubuntu virtual machine with 4 Intel(R) Core(TM) i7-6820HQ CPUs @ 2.70GHz running at 33MHz with 15 GB memory. We record the computation time after doing a warm-up run by executing the tasks 3 times sequentially.

Figure 7 shows the computation time required for performing the reasoning tasks outlined in Algorithm 1. On the x-axis, we report the number of triples for the 16 data sets. On the y-axis, we report the time for completing certain steps of Algorithm 1: (i) map initialization (Line 1), which

Algorithm 1: Current lane identification

```
input : $P_l$: a set of points in lanes,
        $cp_1 \dots cp_n$: a sequence of car positions, $n > 0$
output: $l_1 \dots l_n$: the lanes for the given car positions
1 INITIALIZE()
2 $p_{min}$ = FINDCLOSESTPOINT($P_l$, $cp_1$)
3 $l_1$ = FINDRELATEDLANE($p_{min}$)
4 for $i = 2 \dots n$ do
5     $nbLanes$ = FINDNBLANE($l_{i-1}$)
6     $P_{nbL}$ = COLLECTPOINTS($nbLanes$)
7     $p_{min}$ = FINDCLOSESTPOINT($P_{nbL}, cp_i$)
8     $l_i$ = FINDRELATEDLANE($p_{min}$)
```

includes data import, aggregation and transfer to the high-level ontology; (ii) identification of the lane for the first received position (Lines 2 and 3), and (iii) identification of the lanes for the remaining positions in the trace (Lines 5–8). As an optimization, the search scope is first limited to the points of the neighboring lanes of the previously identified lane (Lines 5 and 6) and then the lane for the closest point to the car position is identified (Lines 7 and 8). Table I provides the detailed results. We group the related actions (importing data and rules) into one transaction for each task, and the computation time is measured for each transaction.

When the size of the data set increases from 23,647 (#1) to 429,702 (#16), the required time for the current lane identification for the first position increases dramatically from 78 ms to 4,811 ms. This is due the fact that the number of points involved in this calculation increases as the number of points in the data set increases. However, the computation time fluctuates slightly between 16 ms to 59 ms as shown in the Fig. 7. This is due to the fact that the number of used points in surrounding lanes of the previous identified current lane remains almost the same, hence the computation time varies by a small margin. The result shows, however, that our approach can reach the required performance.

V. Conclusion and Future Work

In this paper, we presented a knowledge architecture for map data in autonomous vehicles using two levels of ontology abstraction. The presented approach is able to provide: i) a semantic representation of map data, which breaks the dependency of raw data formats; ii) a solution that can be deployed onboard and offboard; iii) a generic interface to access the road environmental knowledge. Based on developed ontologies, we implemented the SmartMapApp prototype. The adequacy of the model is validated over two use cases in a highway scenario: lane change notification and logical inconsistency detection. The results show that the knowledge of the road environment helps to form a consistent environmental model, which facilitates autonomous vehicles to make proper driving decisions.

The road environment modeling in this paper can further be enhanced by integrating the concept of intersections. The current approach can also not yet deal with errors in digital maps, such as road attribute errors or geometric errors and a transaction-based knowledge integrity mechanism still needs to be integrated into the current architecture. We further plan to extend the knowledge layer to cover further map data and offboard map formats and to test the approach in ROS (Robot Operating System) [19], we plan to re-implemented the Java Application SmartMapApp in C++.

References

[1] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous Driving*. Springer, 2016, vol. 10.

[2] L. Zheng, B. Li, H. Zhang, Y. Shan, and J. Zhou, "A high-definition road-network model for self-driving vehicles," *ISPRS Int. J. of Geo-Information*, vol. 7, no. 11, p. 417, 2018.

[3] N. Guarino, D. Oberle, and S. Staab, "What is an ontology?" in *Handbook on ontologies*. Springer, 2009, pp. 1–17.

[4] R. Gayathri and V. Uma, "Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey," *ICT Express*, vol. 4, no. 2, pp. 69 – 74, 2018.

[5] C. Rosse and J. L. Mejino, "The foundational model of anatomy ontology," in *Anatomy Ontologies for Bioinformatics*. Springer, 2008, pp. 59–117.

[6] P. Chahuara, F. Portet, and M. Vacher, "Context aware decision system in a smart home: knowledge representation and decision making using uncertain contextual information," in *4th Int. Workshop on Acquisition, Representation and Reasoning with Contextualized Knowledge (ARCOE-12)*, 2012, pp. 52–64.

[7] C. Villalonga, M. Razzaq, W. Khan, H. Pomares, I. Rojas, S. Lee, and O. Banos, "Ontology-based high-level context inference for human behavior identification," *Sensors*, vol. 16, no. 10, p. 1617, 2016.

[8] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "OWL 2 Web Ontology Language: Primer (2nd Edition)," http://www.w3.org/TR/owl2-primer/, 27 Oct 2009.

[9] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[10] M. Katsumi and M. Fox, "Ontologies for transportation research: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 53 – 82, 2018.

[11] B. Hummel, W. Thiemann, and I. Lulcheva, "Scene understanding of urban road intersections with description logic," in *Dagstuhl Seminar Proc.*, 2008.

[12] R. Regele, "Using ontology-based traffic models for more efficient decision making of autonomous vehicles," in *4th Int. IEEE Conf. on Autonomic and Autonomous Systems (ICAS'08)*, 2008, pp. 94–99.

[13] M. Hülsen, J. M. Zöllner, and C. Weiss, "Traffic intersection situation description ontology for advanced driver assistance," in *IEEE Intelligent Vehicles Symposium*, 2011, pp. 993–999.

[14] M. Hülsen, J. M. Zöllner, N. Haeberlen, and C. Weiss, "Asynchronous real-time framework for knowledge-based intersection assistance," in *14th Int. IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1680–1685.

[15] L. Zhao, R. Ichise, Z. Liu, S. Mita, and Y. Sasaki, "Ontology-based driving decision making: A feasibility study at uncontrolled intersections," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 7, pp. 1425–1439, 2017.

[16] A. Armand, J. Ibanez-Guzman, and C. Zinoune, *Digital Maps for Driving Assistance Systems and Autonomous Driving*. Springer, 2017, pp. 201–244.

[17] Y. Suryawanshi, H. Qiu, A. Ayara, and B. Glimm, "An ontological model for map data in automotive systems," in *IEEE Int. Conf. on Artificial Intelligence and Knowledge Engineering*, 2019, pp. 140–147.

[18] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee, "RDFox: A highly-scalable RDF store," in *Int. Semantic Web Conf.* Springer, 2015, pp. 3–20.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.