Paper number EU-SP0065

Towards a Semantically Enriched Local Dynamic Map

Thomas Eiter[2], Herbert Füreder[1], Fritz Kasslatter[1], Josiane Xavier Parreira[1], Patrik Schneider[1,2]

1. Siemens AG, Austria
2. Institut für Informationssysteme, Technische Universität Wien, Austria

Abstract

With the increasing availability of Cooperative Intelligent Transport Systems, the Local Dynamic Map (LDM) is as a key technology for integrating static, temporary, and dynamic information in a geographical context. Existing ideas do not leverage the full potential of the LDM, since a LDM contains lot of *implicit* information. We aim to provide a *semantically enriched LDM* that applies Semantic Web technologies, in particular ontologies, in combination with spatial stream databases. This allows us to define an enhanced world model, to derive model properties, to infer new information, and to offer expressive query capabilities. We introduce our envisioned architecture which includes a LDM ontology, an annotation and linking framework, static and dynamic data containers, and a query stream-processing component. We also give an overview of three use cases that illustrate the usability and benefits of our approach and provide an initial validation of the use cases in an experimental prototype.

Keywords:

Cooperative ITS, Local Dynamic Map, Semantic Web Technologies

1. Introduction

For Cooperative Intelligent Transport Systems (ITS), the integration of static, temporary, and dynamic information in a geographical context is a crucial feature for a better understanding and processing of traffic scenes. The different ITS systems collect (sensor) data and exchange data by vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), or combined (V2X) communications, which naturally contains temporal data (e.g., traffic light signal phases) and geospatial data (e.g., GPS location) of traffic participants. Motivated by improved safety applications, the authors of the EU SAFESPOT [1] and CVIS [18] projects introduced the concept of the Local Dynamic Map (LDM), which acts as an integration platform to combine static digital maps, also called geographic information system (GIS) maps, with dynamic environmental objects (e.g., vehicles, pedestrians). As shown in Figure 1, the LDM consists of the following four layers:

1. *Permanent static*: The first layer contains static information obtained from GIS map providers and includes roads, intersections, and points-of-interest (POIs);
2. *Transient static:* The second layer extends the static map by further traffic attributes, fixed ITS stations, landmarks, and intersection features like more detailed topological lane data;
3. *Transient dynamic:* The third layer contains temporary regional information like weather, road or traffic conditions (e.g., traffic jams), and traffic light signal phases;
4. *Highly dynamic:* The fourth layer contains dynamic communication nodes as well as other road users detected by V2X messages, in-vehicle sensors like radar and the GPS module.

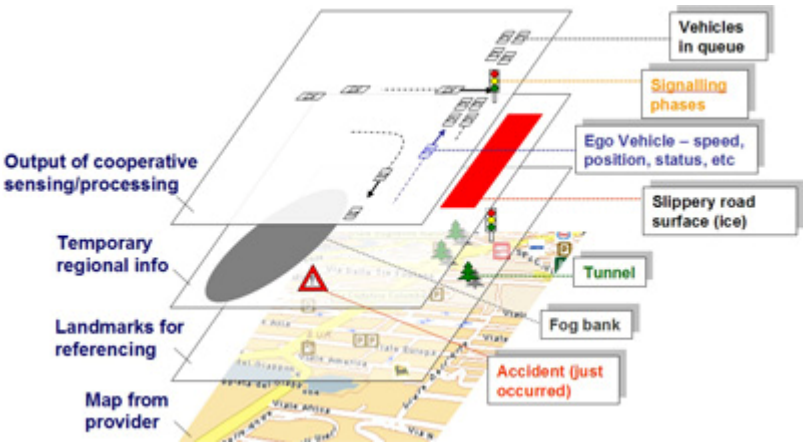

Figure 1 - The four layers of a LDM [1]

We recognize that the LDM is a key technology for data integration in cooperative ITS systems, which is indicated by inital standardization as ETSI [14,15] and ISO [16,17] technical recommendations. Influenced by the ongoing standardization efforts, there is a common understanding that the LDM should include a high-level API and a GIS database with SQL as a query language. Thus, the LDM is a conceptual data store in an ITS station, which integrates sensor data and V2X messages, in particular CAM (Cooperative Awareness Messages), DENM (Decentralised Environmental Notification Messages), MAP (Map Data Messages), and SPaT (Signal Phase and Timing) messages. The authors of [1,21] suggest an object-oriented schema, called *world model*, a topology of geospatial objects, and an object associator, connecting the different objects like individual vehicles and roadside units to the world model. However, the existing ideas do not leverage the full potential of the LDM, which is capable of becoming a more a powerful *integration tool* for sensor data and V2X messages also containing a lot of *implicit* information. An advanced integration can be used to provide new or enhanced functionality for ITS applications. For instance, by combining the lane direction and its type, the trajectory and the role of a vehicle (e.g., ambulance) a wrong-way driver can be detected by querying the stream of V2X messages. By implicit information, we mean all the data which are not directly represented either by V2X messages or by static information from the GIS map. We identified the following list of in particular interesting implicit information in a LDM:

- Part-Whole relations, e.g., *lane2* is part of a *intersection1*
- Spatial relations, e.g., *car1* is on *lane2*
- Connectivity, e.g., *intersection1* is connected to *intersection2* with *road1*
- Functionality, e.g., if *object1* has the same id as *object2*, they are the same.

In this paper, we aim to provide a *semantically enriched LDM* that applies Semantic Web technologies to the standard LDM, which include ontologies (an enhanced world model), spatial stream databases, the related stream processing, and ontology-based data access (OBDA) [23]. Essentially, OBDA is the technique of accessing databases through the ontology by (conjunctive) queries. Adding another layer increases complexity, but we point out that we gain the following advantages from a semantically enriched LDM:

- *World Model:* our notion of a world model is an ontology, which is simply modifiable and extendable. Extensions can be done without altering the database and its relational schema.

- *Model Properties:* the formal models of an ontology and the data have defined properties, which can be used for verification, simplification, and optimization on the conceptual level. For instance, by defining constraints in the ontology (e.g., disjointness between a car and bicycle) inconsistencies in the data can be found.
- *Inference:* OBDA allow us to infer new facts at annotation as well as at query time (e.g., class hierarchies or inverse properties), which reveals implicit information and keeps the amount of stored data small.
- *Expressive Queries:* the queries are posed through the ontology extending the vocabulary beyond database relations. The query language of conjunctive queries (CQ), [1] is simple and yet powerful. Further, by examining the structure of the ontology only certain combination of query atoms are possible.

By semantically enriching the LDM, we highlight the following contributions and related challenges, which we aim to address in this and ongoing work:

- *Modeling*: besides the mentioned ETSI/ISO standards, mobility vocabularies are defined in Schema.org and Mobivoc. [2] Yet, there are no comprehensive ontologies available that capture the LDM and related V2X messages. V2X messages are standardized and thoroughly specified, but they are based on a different modeling language, namely the Abstract Syntax Notation One (ASN.1). The ASN.1 specifications of V2X messages are tree-like and have to be converted to the graph-like structure of ontologies.
- *Annotation*: after the ontology is completed, the different V2X messages and the GIS database have to be mapped to the ontology. The annotation step has to handle static and streaming data in a uniform way and should be easy extendable and maintainable.
- *Query Stream Processing*: the combination of the methods and respective techniques for query stream processing are challenging regarding *performance* and *scalability*. With OBDA, there is a trend into the direction of lightweight query answering over ontologies, thus we can benefit from recent results which improve performance and scalability (cf. [23]). Additional complexity stems from geospatial data and queries evaluated on (stream) database systems, where most implementations are still prototypical. [3]

We proceed as follows. Section 2 describes the state-of-the-art of the LDM. In Section 3 we introduce Semantic Web technologies and stream-processing. Section 4 presents our envisioned architecture to illustrate how Semantic Web technologies can be used for the LDM. In Section 5 we present three use cases to show the benefits of a semantically enriched LDM. Section 7 concludes with possible future works and refinements.

2. State-of-the-art Definitions of the LDM

In this section, we have a closer look at state-of-the-art efforts regarding the LDM.

SAFESPOT Project. The authors of the SAFESPOT project initiated the term and definition of the LDM in work package D 7.3.1 [1]. They recognized that the data model “has a hierarchical structure using associations between classes to describe their relationships”.

[1] CQs are a lean representation of SQL select-project-join queries: `q(x) = MAPLane(x) ∧ isIngress(x,TRUE)` is rewritten into `SELECT a.x FROM MAPLane AS a, isIngress AS b WHERE a.x = b.x AND b.y = TRUE`
[2] cf. http://schema.org/ and http://www.mobivoc.org/
[3] e.g., PipelineDB on https://www.pipelinedb.com/

However, they dropped an object-orient model in favor of a relational model tailored to a Relational Database Management System (RDBMS) due to performance concerns. The authors also suggested two implementations based on commercial GIS tools. PG-LDM is developed by Tele Atlas and built on top of PostGIS, which is a good choice, if used on general-purpose systems and complex spatial queries are desired. NAVTEQ-LDM is built by Navteq and uses SQLite as the background RDBMS already targeting embedded systems, where SQLite is widely available. The authors also suggest a database schema representing the four layers. The schema includes the different groups of tables including static features, moving objects, conceptual objects, and relationships. Finally, they defined an API that supports custom functions (e.g., *getLanesForRoadElement*) and a direct SQL query interface.

Research Prototypes. Netten et al. [21] introduced DynaMap, an extended architecture for LDM and recognized that the focus of previous work on the LDM is *car-centric* and put the focus of the LDM to roadside ITS stations. They defined a new architecture which includes data sources, a world model, world objects, and data sinks. World objects are created by the world object associator based on the streamed input from the different data sources which include V2X messages and sensors. The world model resembles an ontology and defines the relationships between all the objects including their hierarchical relations and current states including a history of them. They also recognize that each object has a reference position, which relates to a spatial topology. Koenders et al. [19] developed an "open dynamic map" where they point out that the LDM cannot store all objects and their data points permanently, thus introducing a custom stream processing by deleting objects which are too far from to the ITS station. They designed their own relational schema having tables for areas, objects, and roads including a spatial topology. They also provide additional functions to the LDM, which include map-matching and a security layer. Shimada et al. [24] implemented the initial (RDBMS-centric) approach by SAFESPOT again and evaluated it in a complex collision detection use case. For the evaluation, a traffic simulation tool was used to generate V2X messages for different numbers of vehicles. The authors also recognized that the GIS maps and tools do not need to be commercial and extracted the road graph from OpenStreetMap (OSM).

ETSI/ISO Standards. Initial standardization happened by the ETSI TR 102 863 (V1.1.1) [14] report, where a LDM is defined as "a conceptual data store located within an ITS station … containing information which is relevant to the safe and successful operation of ITS applications". The report positions the LDM into the facilities layer of the ITS station reference architecture. Then it connects the four layers with possible ITS applications. For instance speed limitation is defined in the third layer and can be used for co-operative speed management. The report also recognizes that the LDM architecture is made of a management and a data store, which can be accessed through an API with three interfaces, namely *AF-SAP* (Applications), *NF-SAP* (Networking), and *SF-SAP* (Security). It also addresses the topic of how the LDM can be linked to the road network of a static GIS map (the first layer) called "dynamic location referencing". Within the ETSI EN 302 895 (V1.1.0) final draft [15], the work of the previous report was extended with new functionalities, introducing LDM *Data Objects*, which are compositional data structures, and LDM *Data Providers/Customers*. Also a new interface for

LDM services and maintenance was defined. Via the interface Data Objects can be fetched with SQL-like filtering and selection statements. We see a semantically enriched LDM as the fusion of LDM Data Objects and Providers. With an international focus, the ISO/TS 17931:2013 [16] and ISO/TS 18750:2015 [17] reports define comparable standards to ETSI, which include a LDM architecture, data models, and its embedding into the ITS architecture.

3. Background Technology and Methods

In this section we give a brief introduction into the methods and technologies that we envision to use for achieving a semantically enriched LDM.

Semantic Web Technologies. Semantic Web technologies provide a common framework for sharing and reusing data across boundaries. We refer to the seminal article of Berners-Lee et al. [8] for an outline of the ideas and to the Semantic Web stack[4] for an architectural overview of it. The Resource Description Framework (RDF)[5] serves as a flat, graph-based unified data model which is based on URIs as identifiers for objects and relations. An RDF graph is represented by triples (S,P,O) of a subject S, a predicate P, and an object O. Ontologies are used for modeling knowledge domains, by expressing relations between terms with a restricted vocabulary and by modeling them as class hierarchies. In the Semantic Web context, OWL[6] plays a central role as the standard modeling language of ontologies with its (formal) logical underpinning of Description Logics (DL) [5]. The vocabulary of a DL consists of *objects* (called *individuals*), *classes* (called *concepts*), and *properties* (called *roles*). Furthermore, a *knowledge base* (KB) consists of a *terminological box* (TBox), which contains axioms about relations between classes and properties, and an *assertional box* (ABox), which contains factual knowledge about individuals by the following assertions represented as N-triples[5] (cf. [5]):

- Class assertions: `<id1> <type> <Class> .` states object `id1` belongs to `Class` *(*e.g., `Car`);
- Object property assertions: `<id1> <property1> <id2> .` states object `id1` is related by `property1` to object `id2` *(e.g.,*`lane1 isPartOf intersection2`*)*;
- Data property assertions: `<id1> <property2> value .` states object `id1` has the `property2` with a numeric `value` (e.g., `car1 hasSpeed 20`*)*.

In the light of data-intensive applications, there is a trend to move from the expressive OWL 2 towards more scalable and tractable fragments called OWL 2 Profiles.[6] These research efforts have been focused on efficient query answering techniques over lightweight ontology languages, such as OWL2 QL also called *DL-Lite* [10] and *EL++* [4] families. Conjunctive query evaluation over OWL2 QL ontologies can be delegated, by first-order query rewriting, to a RDBMS, which facilitates scalable query processing. Ontologies modeled with OWL2 QL are well suited for defining the conceptual level. The Ontop system [23] is an example of an ontology-based data access (OBDA) system, where a global schema is defined as a OWL2 QL ontology, and the source schemas are mapped to the global schema by SQL queries.

Stream-Processing and Stream-Reasoning. Relational stream processing has been investigated for long, e.g., the TelegraphCQ system. An important step was CQL [3] with the design goals of a clear syntax based on SQL-99 and an abstract (relational) semantics. The

[4] A recent version is at https://commons.wikimedia.org/wiki/File:Semantic_web_stack.svg

[5] W3C recommendation for the format at http://www.w3.org/TR/rdf-primer/ and http://www.w3.org/TR/n-triples/

[6] W3C recommendation at http://www.w3.org/TR/owl2-primer/ and http://www.w3.org/TR/owl2-profiles/

authors defined *streams* and *relations* as data types and the following operators that map between them: namely (1) *stream-to-relation*, (2) existing *relation-to-relation*, and (3) *relation-to-stream*. For (1), they introduced *time-based sliding*, *tuple-based sliding*, and *partitioned window* operators to select tuples from a stream. For (3), they defined *insert stream*, *delete stream*, and *relation stream* operators. Furthermore, they developed a method for query execution and benchmarked a prototype implementation on a highway tolling use case to show the simplicity and applicability of their approach. With *linked stream processing* the streams are lifted to a "semantic" level by representing them as triples and connecting them to ontologies. These approaches, i.e., CQELS [20] and C-SPARQL [6], need either an annotation step or a data wrapper to convert the streamed tuples into triples. Often the same window operators as in relational stream processing are used. Further expressivity is provided by *stream reasoning*, which allows for knowledge intensive processing by extending the window operators with temporal relations (e.g., valid until) between different windows. Roughly, stream reasoning approaches can be split into *rule-based* approaches like LARS [7] and ETALIS [2]; and *non-rule-based* approaches like the work of Cayuga [9] and Özçep et al. [22]. The later is of particular interest as it combines OBDA and stream processing.

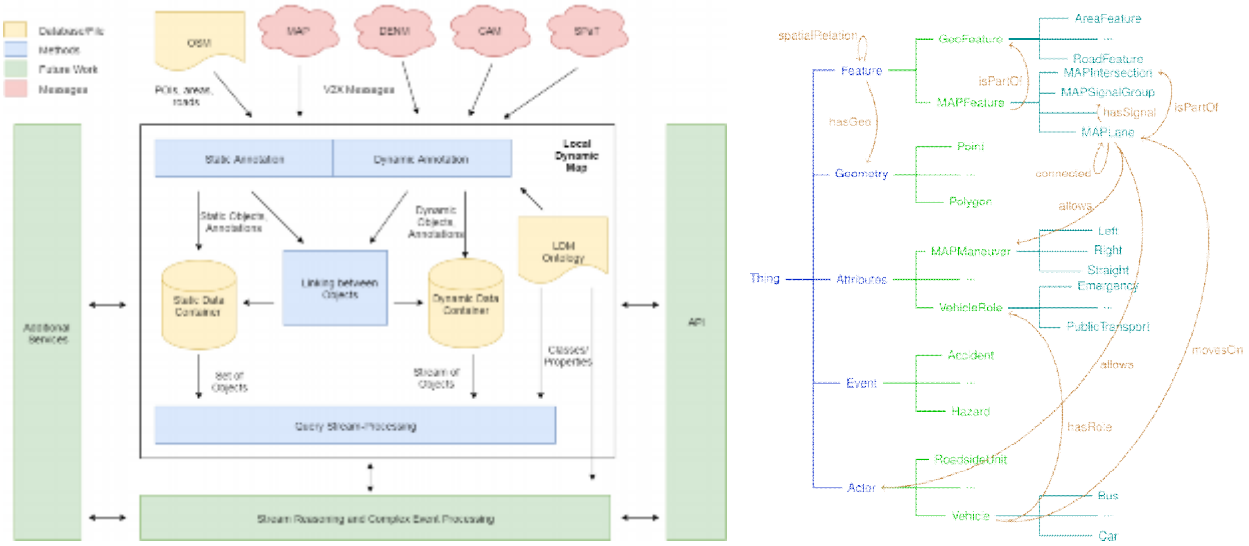

Figure 2 - (a) System architecture and (b) LDM ontology (partial rendering)

4. Architecture of a Semantically Enriched LDM

Based on the mentioned technologies, we introduce our envisioned architecture in Figure 2a to illustrate how Semantic Web technologies can be used for the LDM. Our approach is related to DynaMap, but our emphasis is more on stream processing with ontologies, whereas in DynaMap an object-oriented data model and processing techniques were used. We point out that every component of the architecture is a complex topic by itself, thus we sketch the main ideas and show how the architecture can be implemented based on our existing work in spatial query answering [11,12,13]. Except the ontology and its interaction with the other components, we will refine the initial components and work out technical details in future work.

LDM Ontology. The LDM ontology[7] is represented using the W3C standard OWL2 QL [10] and shown partially in Figure 2b. We follow a layered approach starting at the bottom with a simple separation between the following classes and properties:

- V2X features (`MAPFeature`) which model the content of the MAP topology;
- geospatial features (`GeoFeature`);
- geometrical representations of features (`Geometry`);
- the main actors (`Actor`) involved in a transport scene;
- events (`Event`) which are related to the transport domain;
- partonomies (`isPartOf`), spatial relations (`intersects`), connectivity (`connected`), and other properties like `isAllowed`, `hasRole`, and `isManaged`.

Geospatial features define POIs, roads, and areas. These classes are also linked to GeoOWL and GeoNames for a embedding in existing ontologies.[8] `MAPFeature` is a domain specific modeling of the MAP topology and includes the classes which describe the details of an intersection including its lanes, roadside units, allowed maneuvers, traffic lights, etc. The `Actor class` (e.g., amulance) is linked by the `hasRole` property to lanes, etc. It is split into `Person`, `Vehicle`, and `RoadsideUnit` which play different roles in a transport environment.

Annotation framework (AF). The AF automatically receives the V2X messages and extracts the raw V2X message data and connects the elements of a message to the ontology, which then is added to the data containers. The AF has the following steps:

1. Creating a unique ID for each V2X message and adding the ID and the raw message body to the static or dynamic data container. By keeping the message content, additional assertions could be derived at query evaluation;
2. Creating the static and dynamic ABox assertions based on the ID of the objects by applying *Datalog* rules as in [13] on the filtered message elements.

For initial experiments, we automatically annotate CAM, SPaT, and DENM messages, yet annotate MAP messages manually. The AF can take advantage of our work on the city-bench benchmark creation tool [13], which combines Datalog rules with simple extract, transform, and load (ETL) features. Each rule is a mapping from a *source* to *target* with an optional *transformation* step in between. The sources include geospatial data sources like OSM, while the target is designed to produce ABox assertions. For instance, the following rules create class and object property assertions for all police stations in a city, where $\texttt{Geo}_{\texttt{Point}}$ and $\texttt{Tag}_{\texttt{Police}}$ extracts the OSM points that are tagged as "police":[9]

$$\texttt{Geo}_{\texttt{Point}}(x) \wedge \texttt{Tag}_{\texttt{Police}}(x) \rightarrow \texttt{PoliceStation}(x)$$

$$\texttt{Geo}_{\texttt{Point}}(x) \wedge \texttt{Tag}_{\texttt{Police}}(x) \wedge \texttt{Geo}_{\texttt{Polygon}}(y) \wedge \texttt{Tag}_{\texttt{City}}(y) \wedge \texttt{Inside}(x, y) \rightarrow \texttt{hasPoliceStation}(y, x)$$

Further, the atom $\texttt{V2XMAP}_{\texttt{Lane}}$ is used to extract different incoming V2X messages and allows us to filter different sections of a MAP message, e.g., the lanes.

Linking Framework (LF). The LF computes and stores links between objects that are not directly represented in the data. The linking could be computed off- or online and uses the same

[7] A draft is available at http://www.kr.tuwien.ac.at/research/projects/myits/LocalDynamicMapITS-v0.1-Lite.owl
[8] cf. http://www.w3.org/2005/Incubator/geo/XGR-geo/ and http://www.geonames.org/ontology/
[9] `PoliceStation(x)` and `hasPoliceStation(y, x)` are legible renderings of the N-Triples `x <type> <PoliceStation> .` and `y <hasPoliceStation> x .`

rules and sources as the AF. For the LF, spatial relations are the main focus, where we follow the standard for RDBMS called Dimensionally Extended Nine-Intersection Model (DE-9IM), which supports relations like `Touches`, `Intersects`, and `Disjoint`. One important link is the information about lanes that are adjacent to each other. The following rule collects from a MAP message the lanes that are adjacent:

$$\text{Geo}_{\text{Polygon}}(x) \wedge \text{V2XMAP}_{\text{Lane}}(x) \wedge \text{Geo}_{\text{Polygon}}(y) \wedge \text{V2XMAP}_{\text{Lane}}(y) \wedge \text{intersects}(x,y) \rightarrow \text{adjacentLane}(x,y)$$

Another important link is the connection of a MAP intersection to the intersection in the OSM road graph. For this, more advanced computation methods are needed, since different geospatial objects (e.g., points and polygons) have to be mapped to each other.

Static Data Container (SDC). The SDC is a data container for spatio-relational data which never or infrequently changes thus keeping the data of the first and second layer. The following database relations are kept in the SDC: The database relations `Point`, `Polygon`, `Lines`, and `Roads` representing different types of geospatial objects of an OSM map. For instance the relation `Point` includes different POIs like petrol stations or bus stops; The database relation `MAP` representing MAP messages and the elements of a road intersection, e.g., lanes, crossing, etc. After annotation (and conversion) of a message, most of the objects have a geometrical representation (e.g., polygons); The database relations for the static ABox assertions that connect OSM and LDM objects to the ontology. The ABox is represented by the relations for class and property assertions.

Objects of the first layer are initially added from a geographically restricted instance of OSM (e.g., region of Vienna) and stored in the SDC. Objects of the second layer are the elements of a road intersection and are either fixed (for an ITS station) or incoming MAP messages (for a vehicle), since the vehicle might pass several road intersections. For both cases the AF is used to add the objects (e.g., lanes) taken from one or more MAP messages to the SDC and link them to the ontology by class and property assertions.

Dynamic Data Container (DDC). The DDC is a stream data container and keeps the dynamic data of the third and fourth layer. The DDC foresees push- and pull-based stream processing. Initially we focus on pull-based processing, which adds the requirement that each raw message and each assertion has a timestamp assigned. Then, we use the timestamp on query time for filtering with a window operator. The following database relations are kept in the DDC:
The database relations `CAM`, `DENM`, and `SPaT`, which store the raw message data , e.g., vehicle positions, signal phases, etc.; The database relations for the dynamic ABox assertions which link the dynamic objects to the ontology. Note that class assertions are only static. We will also investigate in the future *continuous views* over *data streams* for push-based processing (see below) in stream databases as PipelineDB.

Query Stream-Processing. This component is central to the usage of a semantically enriched LDM, since the other components have to be connected at this point. We assume that the annotation and linking for static and dynamic data occurred previously. There are two types of query evaluation: *pull-based queries*, which evaluate at a single point in time (i.e., the query time) the state of a LDM; *push-based queries*, which evaluate the query continuously and return the ongoing changes of a LDM. We outline our method for pull-based queries and leave the more complex push-based querying for future work. For pull-based queries, we again take

advantage of our previous work on OBDA for spatial RDBMS [11]. We extend the approach of [11] with a window operator that selects an actual snapshot of the DDC relations. This leads to the following three stages:

1. Extract the join tree from the input CQ, where each node represents either an *ontology atom*, e.g., `MotorVehicle(x)`, *spatial atom*, e.g., `intersects(x,y)`, or *stream atom*, e.g., `position(x,y);`
2. Evaluate the join tree bottom up to create a set of CQs called union of CQs (UCQ) and distinguish between (a) ontology, (b) spatial, and (c) stream nodes:
   (a) Apply OBDA rewriting [10,23] and create for each ontology/stream atom one UCQ;
   (b) Rewrite the spatial atoms into corresponding SQL functions;
   (c) Apply either (a) or (b) and add the window operator with a time inverval.
3. Merge all UCQs to one large UCQ and rewrite it into SQL select-project-join queries, which are then evaluated over the SDC and DDC.

In Step 1, the input CQ has to be acyclic to keep the computational complexity low, e.g., `q(x,y,z) = hasPart(x,y) ∧hasPart(y,z) ∧hasPart(z,x)` is not allowed. For Step 2 and 3, an alternative strategy would be the partially evaluation of UCQs up to a spatial node to keep intermediate results, which can be evaluated internally using spatial functions. Further, the time interval for a window operator is included in the CQ, e.g., $\texttt{speed}_{\texttt{(Range 30s)}}\texttt{(y,r)}$.

API. The LDM is defined as a data integration platform which provides services to external applications. We aim to support different types of APIs on top of our approach. First we aim to support the standard API requirement by the ETSI TR 102 863. We assume that the *SF-SAP* is handled by the ITS station so it is not in the scope of our work. As described in the standard, the *NF-SAP* interface connects the LDM to the communication functions of the ITS station and receives the V2X messages, which are then forwarded to the AF and LF.

5. Use Cases and Experiments

In this section, we give an overview of three use cases that illustrate the usability and benefits of our approach. The use cases are related to our existing roadside ITS station on a road intersection at Handelskai, Vienna depicted in Figure 3. The ITS station receives any V2X message, and is capable to send SPaT (traffic light signals) and MAP (intersection topology) messages. The first two use cases are concerned with static data, where the first use case checks the consistency of a MAP. In the second use case, we show how an integration of OSM with the MAP intersection can be done for collecting information about the broader surroundings (e.g., parking areas). The third use case is concerned with dynamic data arising from vehicle movements (CAM) and signal phases of the traffic lights (SPaT).

Use Case 1 (UC1) - Lane Consistency. The focus of this use case is system driven and shows how lane *inconsistency* in a MAP message can be detected. For this we identify two possible inconsistencies and give the queries to detect them. To detect them, the results of the queries listed below have be compared to a base query $\texttt{q}_1\texttt{(x) = MAPLane(x) isPartOf(x,y) ∧ MAPIntersection(y)}$ that finds all lanes on an intersection. Another approach could include negation (i.e., `not`) in the queries; this would lead to an extension of the common OBDA methods. The following queries show consistency checks, where the query atom `connected(x,y)` is extracted from MAP messages:

1. Ingress lanes x for cars should have at least one outbound connection to an outgress lane:

$q_2(x) = \text{MAPLaneInForMotor}(x) \wedge \text{connected}(x,y) \wedge \text{MAPLaneOutForMotor}(y) \wedge \text{isPartOf}(x,z) \wedge \text{MAPIntersection}(z)$

2. Each ingress lane x should allow a least one maneuver, i.e., turn left:

$q_3(x) = \text{MAPLaneIn}(x) \wedge \text{connected}(x,) \wedge \text{allowsManeuver}(x,y) \wedge \text{MAPManeuver}(y) \wedge \text{isPartOf}(x,y) \wedge \text{MAPIntersection}(x)$

The query rewriting of OBDA applies several steps of inference, therefore these queries would be hard to formulate manually in SQL. For instance `MAPLaneForMotor` is a sub-class of `MAPLane` and `MAPLaneInForMotor` is a compound of `MotorVehicle`, which again has the sub-classes `Bus`, `Car`, and `Truck`. The property `connected` is the inverse of `isConnected`, which means that the instances are taken from both properties.

Use Case 2 (UC2) - OSM Integration. For the OSM Integration, we show that by merging V2X messages and OSM maps, more detailed information of the surroundings are obtained, which leads to a better situation awareness of the ITS station. This is achieved by combining the OSM street graph and the MAP messages of the nearby intersections. UC2 opens up the possibility to evaluate the program of the ITS station's traffic light controllers and give suggestions for traffic flow optimizations also taking into account the neighboring intersections and the ITS station. The queries show that following information from the surroundings can be retrieved:

1. Find all parking areas y from OSM that are located close to our intersection:

$q_4(y) = \text{MAPIntersection}(x) \wedge \text{isEgo}(x,\text{TRUE}) \wedge \text{hasGeo}(x,u) \wedge \text{near}(u,v) \wedge \text{hasGeo}(y,v) \wedge \text{ParkingArea}(y)$

2. Find all the neighboring intersections z that are connected by a main road in OSM:

$q_5(z) = \text{MAPIntersection}(x) \wedge \text{isEgo}(x,\text{TRUE}) \wedge \text{hasGeo}(x,u) \wedge \text{intersects}(u,v) \wedge \text{hasGeo}(y,v) \wedge \text{PrimaryRoute}(y) \wedge \text{intersects}(v,w) \wedge \text{hasGeo}(z,w) \wedge \text{MAPIntersection}(z)$

The spatial relations `near` and `intersects` are evaluated at query time, since we have a quadradic combination of OSM objects for each relation. However, the street graph and buildings are annotated and extracted from OSM offline.

Use Case 3 (UC3) - Safety Application. The focus of the dynamic use case is driven by the aim of improving road safety on heavily frequented intersections by (a) detecting amulances and (b) stop violations (e.g., car crossing at red light) by normal vehicles. For this, the integration of all the V2X messages is needed. We use the LDM to detect, whether (a) an ambulance is on one of the lanes and (b) a vehicle is moving above a certain speed on a lane whose current signal phase is on stop. This is expressed by the following queries:

1. Find all ambulances y that are currently on our intersection:

$q_6(z) = \text{MAPIntersection}(x) \wedge \text{isEgo}(x,\text{TRUE}) \wedge \text{isPartOf}(x,y) \wedge \text{MAPLane}(y)\ \text{hasGeo}(y,u) \wedge \text{contains}(u,v) \wedge \text{position}_{(\text{Range 30s})}(z,v) \wedge \text{MotorVehicle}(z) \wedge \text{hasRole}(z,\text{AmbulanceRole})$

2. Find all cars y and lanes x that are moving above 30kph and violate the signal phase stop:

$q_7(x,y) = \text{MAPLane}(x) \wedge \text{hasGeo}(x,u) \wedge \text{contains}(u,v) \wedge \text{position}_{(\text{Range 30s})}(y,v) \wedge \text{Car}(y) \wedge \text{speed}_{(\text{Range 30s})}(y,r) \wedge (r>30) \wedge \text{isManaged}(x,z) \wedge \text{MAPSignal}(z) \wedge \text{hasState}_{(\text{Range 90s})}(z,\text{Stop})$

Since we have dynamic data, the query atoms $\text{position}_{(\text{Range 30s})}(y,v)$, $\text{speed}_{(\text{Range 30s})}(y,r)$, and $\text{hasState}_{(\text{Range 90s})}(z,\text{Stop})$ specify a time-based sliding window of `30s` resp. `90s`. of the DDC.

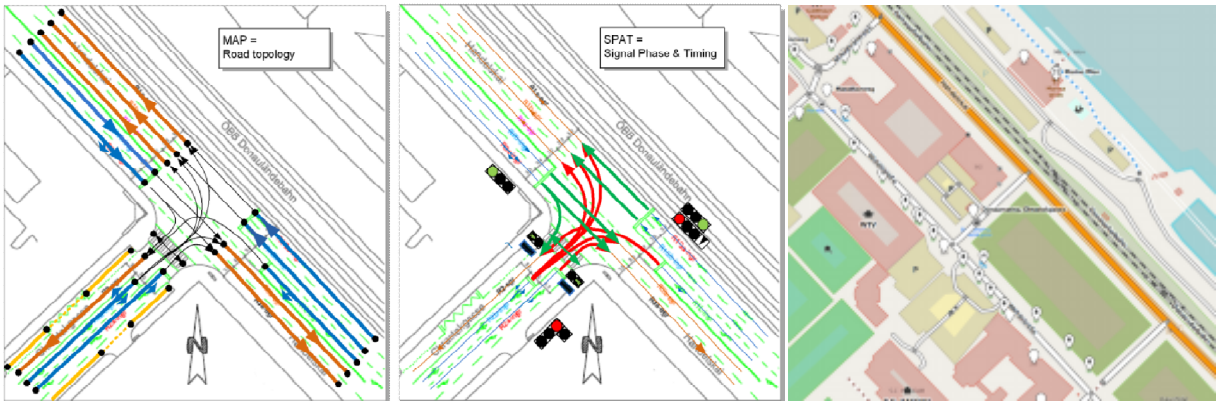

Figure 3 - MAP topology, the related SPaT messages, and OSM of Handelskai

Initial Implementation and Experiments.[10] We give an overview of an early experimental prototype and an initial validation of the use cases on data from the roadside ITS station. We slightly adapted our prototype for spatial query answering with OWL2 QL [11], which is developed in Java 1.6 and uses PostGIS 1.5.1 (for PostgreSQL 9.0) as the spatial-extended RDBMS. For our intersection, the query evaluation times of UC1 and UC2 were below 2s without optimization on a Mac OS X 10.6.8 system with an Intel Core i7 2.66GHz and 8 GB of RAM. For UC1, we found no inconsistency and the OBDA rewritting had a size of 18 atoms for q_1, 143 atoms for q_2, and 88 atoms for q_3. In UC2, we applied the annotations using [12] for `ParkingArea` (resp. `PrimaryRoute`) which created 3436 (resp. 3973) annotated objects for OSM Vienna. The rewritings were small with five ontology resp. one spatial atom(s) for q_4 and seven ontology resp. two spatial atoms for q_5. UC3 is work in progress, but we provide the annotation rules and tools which are needed for processing CAM and SPaT messages.

6. Conclusion and Future Work

In this paper, we presented an extension of the LDM with Semantic Web technologies, which gives us the possibility to define a world model, i.e., the ontology, an expressive and simple query language, derived model properties, and the capabilities to infer new facts. Our envisioned architecture is designed to show how these technologies can be applied in the LDM. The architecture consists of a LDM ontology, an annotation and linking framework, static and dynamic data containers, and foresees a query component over streams. We provide the ontology, and show how the architecture can be implemented based on existing work in spatial query answering [11,13]. Further, we developed the use cases lane consistency, OSM integration, and safety application to show the usability and benefits of our design.

Future research is directed to extending the components of the framework. In particular, the annotation and linking framework as well as the query component could be elaborated. The later by realizing push-based querying with our *Clipper* system [12] using CQL [3]. Also further investigations regarding expressivity and efficiency are needed. The three use cases are good starting points to evaluate the framework and its implementation in a larger test setting including real-life benchmarks. Finally, future work would be directed towards methods for data aggregation, complex event detection, and model-based diagnosis.

[10] Tools, annotations, queries are on https://github.com/ghxiao/city-bench/benchmarks/its2016

References

1. L. Andreone, R. Brignolo, S. Damiani, F. Sommariva, G. Vivo, S. Marco (2010). D8.1.1 – SAFESPOT Final Report. Technical report. Available at http://www.transport-research.info/ Upload/Documents/201303/20130329_130257_17414_D8.1.1_Final_Report__Public_v1.0.pdf.
2. D. Anicic, P. Fodor, S. Rudolph, R. Stühmer, N. Stojanovic, R. Studer (2011). Etalis: Rule-based reasoning in event processing. In: Reasoning in Event-Based Distributed Systems 2011, SCI vol. 347, Springer, 99-124.
3. A. Arasu, S. Babu, J. Widom (2006). The CQL continuous query language: semantic foundations and query execution. VLDB Journal, 15(2), 121-142.
4. F. Baader, S. Brandt, C. Lutz (2005). Pushing the EL Envelope. In: Proc. of IJCAI 2005, 364-369.
5. F. Baader, I. Horrocks, U. Sattler (2009). Description logics. In: Handbook on Ontologies, Springer, 21-43.
6. D. F. Barbieri, D. Braga, S. Ceri, M. Grossniklaus. (2010). An execution environment for c-sparql queries. In: Proc. of EDBT 2010, ACM, 441–452.
7. H. Beck, M. Dao-Tran, T. Eiter, M. Fink (2015). LARS: A Logic-based Framework for Analyzing Reasoning over Streams. In: Proc. of AAAI 2015, AAAI Press.
8. T. Berners-Lee, J. Hendler, O. Lassila (2001). The semantic web. Scientific American 284(5), 34-43.
9. L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, W. White. (2007). Cayuga: a high-performance event processing engine. In: Proc. of 2007 ACM SIGMOD, ACM, 1100-1102.
10. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati (2007). Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. Journal of Automated Reasoning 39(3), 385-429.
11. T. Eiter, T. Krennwallner, P. Schneider (2013). Lightweight Spatial Conjunctive Query Answering using Keywords. In: Proc. of ESWC 2013, LNCS, vol. 7882, Springer, 243-258.
12. T. Eiter, M. Ortiz, M. Simkus, T.K. Tran, G. Xiao (2012). Query Rewriting for Horn-SHIQ plus Rules. In: Proc. of AAAI 2012, AAAI Press.
13. T. Eiter, J. Z. Pan, P. Schneider, M. Simkus, G. Xiao (2015). A Rule-based Framework for Creating Instance Data from OpenStreetMap. In: Proc. of RR 2015, LNCS, vol. 9209, Springer, 93-104.
14. ETSI TR 102 863 (V1.1.1) (2011). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and Guidance on Standardization.
15. ETSI EN 302 895 (V1.1.0) (2014). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM).
16. ISO/TS 17931:2013 (2013). Intelligent transport systems - Extension of map database specifications for Local Dynamic Map for applications of Cooperative ITS.
17. ISO/TS 18750:2015 (2015). Intelligent transport systems - Cooperative systems - Definition of a global concept for Local Dynamic Maps.
18. P. Kompfner (2010). D.CVIS.1.3 – Final Activity Report Period: 01/02/2006 to 30/06/2010. Technical report, CVIS (FP6-2004-IST-4-027293-IP), Available at http://www.cvisproject.org/ download/Deliverables/DEL_CVIS_1.3_FinalActivityReport_PartII_PublishableSummary_V1.0.pdf
19. E. Koenders, D. Oort, K. Rozema (2014). An open Local Dynamic Map. In: Proc. of ITS European Congress 2014, ERTICO - ITS Europe.
20. D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, M. Hauswirth (2011). A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In: Proc. of ISWC 2011, LNCS vol. 7031, Springer, 370-388.
21. B. Netten, L.J.H.M. Kester, H. Wedemeijer, I. Passchier, B. Driessen (2013). DynaMap: A Dynamic Map for road side ITS stations. In: Proc. of ITS World Congress 2013, Intelligent Transportation Society of America.
22. Ö. L. Özçep, R. Möller, C. Neuenstadt (2014). A Stream-Temporal Query Language for Ontology Based Data Access. In: Proc. of Description Logics 2014, 696-708.
23. M. Rodriguez-Muro, R. Kontchakov, M. Zakharyaschev (2013). Ontology-Based Data Access: Ontop of Databases. In: Proc. of ISWC 2013, LNCS vol. 8218, Springer, 558-573.
24. H. Shimada, A. Yamaguchi, H. Takada, K. Sato (2015). Implementation and Evaluation of Local Dynamic Map in Safety Driving Systems. Journal of Transportation Technologies, 5, 102-112.