Paper number EU-SP0065

Towards a Semantically Enriched Local Dynamic Map

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

1. Institut für Informationssysteme, Technische Universität Wien, Austria
2. Siemens AG, 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 an LDM contains streaming data and varying *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, derive model properties, infer new information, and offer expressive query capabilities over streams. We introduce our envisioned architecture which includes an LDM ontology, an annotation and linking framework, data containers, and a stream query answering component. We also sketch three application scenarios that illustrate the usability and benefits of our approach and provide an initial validation of the scenarios 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 the understanding and processing of complex traffic scenes. Different ITS systems collect (sensor) data and exchange it by vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), or combined (V2X) communications, which naturally contains temporal (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 SAFESPOT [1] and CVIS [17] 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 (representing e.g., vehicles or 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 maps that include 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 information, mainly V2X messages of road users including their GPS position and speed.

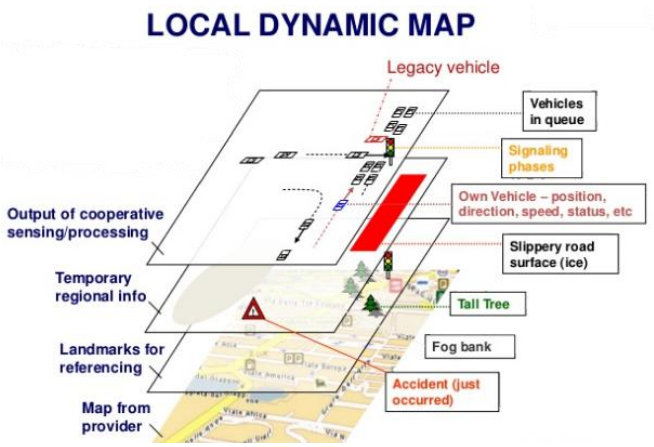

Figure 1 - The four layers of an LDM [1]

We recognize that the LDM is a key technology for data integration in cooperative ITS systems, which is indicated by its initial standardization as ETSI [13,14] and ISO [15,16] technical recommendations. Influenced by the ongoing standardization efforts, there is a common understanding of the LDM that it should include a high-level API, a GIS database (DB), and 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 (Decentralized Environmental Notification Messages), MAP (Map Data Messages), and SPaT (Signal Phase and Timing) messages. The authors of [1,20] 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 address the "streaming" nature of the data and leverage the full potential of the LDM, which is capable of becoming a powerful *integration tool* for sensor data and V2X messages in general, allowing complex scene recognition by utilizing *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., ambulances) 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 an LDM:

- Part-Whole relations, e.g., a lane is part of an intersection;
- Spatial relations, e.g., a car is crossing a stop lane;
- Connectivity, e.g., intersection is connected to intersections via a road;
- Functionality, e.g., if two objects have the same id, 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 DBs, the related stream processing, and ontology-based data access (OBDA) [22] for connecting the ontology with the DB. Essentially, OBDA is the technique of accessing DBs 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 DB 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 inconsistencies in the data can be found (e.g., disjointness between a car and bicycle).

- *Inference:* OBDA allows us to infer new information at query time (e.g., class hierarchies), 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 DB 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 DB have to be mapped to the ontology. The annotation step has to handle static and streaming data in a uniform way and should be easily extendable and maintainable.
- *Stream Query Answering*: the combination of the methods and respective techniques for query answering (QA) over streams 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. [10,22]). Additional complexity stems from geospatial data and the queries evaluated on stream DB systems as PipelineDB,[3] where most implementations are designed towards efficiency but not complex query evaluation using ontologies.

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 application scenarios to show the benefits of a semantically enriched LDM. Section 6 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". 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

[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] https://www.pipelinedb.com/

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 DB 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.

ETSI/ISO Standards. Initial standardization happened by the ETSI TR 102 863 (V1.1.1) [13] report, where an 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 and 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 [14], 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 an extension of LDM Data Objects and Providers. With a more international focus, the ISO/TS 17931:2013 [15] and ISO/TS 18750:2015 [16] reports define comparable standards to ETSI, which include an LDM architecture, data models, and its embedding into the ITS architecture.

Research Prototypes. Netten et al. [20] introduced DynaMap, an extended architecture for an 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 novel 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. [18] 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 the ITS station. They designed their own relational schema having tables for areas, objects, and roads including a spatial topology. They also provided additional functions to the LDM, which include map-matching and a security layer. Shimada et al. [23] implemented the initial (RDBMS-centric) approach by SAFESPOT again and evaluated it in a complex collision detection application scenario. For the evaluation, a traffic simulation tool was used to generate

V2X messages for different numbers of vehicles. The authors also recognized that GIS maps and tools can be open-source and extracted the road graph from OpenStreetMap (OSM).

Current Shortcomings. Above efforts are already mature and allow an elaborate usage of the LDM. However, these efforts are limited if used in a complex changing environment with vehicles updating at frequent intervals. We believe the following shortcomings are still present:

- *Database-centric:* besides DynaMap, all other efforts have a database-centric model of the LDM using a static schema, where the LDM objects are directly mapped to relational tables. New types of objects need a modification of the schema, which makes it harder to add new domains, e.g., traffic regulations. Further the DB schema cannot capture class hierarchies and the dependencies between the different objects, since it is not graph-based.
- *Efficient streaming:* except DynaMap, other efforts are designed to work on top of a GIS DB (e.g., PostGIS) and neglect the streaming nature of the LDM data, since it should allow real-time queries over large amounts of data "in-stream", i.e., without storing. Stream processing needs a clear data model (point-based vs. interval-based) and a defined query language that supports window operators (sliding or fixed size), stream joins, and aggregation over streams (e.g., average speed over 30s). These features are either entirely missing or considered in external components.
- *Sound integration:* the integration of all layers and the model of a complex intersection could lead to incomplete or inconsistent data, e.g., MAP has wrongly connected lanes. RDBMS allow integrity constrains, however they are only implicit and do not suffices for all integrity cases (e.g., connectivity).

DynaMap already addressed to some extend the above shortcomings but differs from our work in three points: (a) our *world model* is based on a standard language using ontologies, thus existing approaches and algorithms can be applied directly; (b) the streaming is based on "monitors" which appear to be separated from the query language, which makes optimization and integration harder; and (c) checking inconsistency is not an aim of DynaMap. Thus, our emphasis is more on query answering over streams with ontologies, whereas in DynaMap an object-oriented data model and custom processing techniques are used.

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

[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/

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 a `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 SQL 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 [22] is an example for an ontology-based data access (OBDA) system, where a global schema is defined as an 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 authors defined *streams* and *relations* as data types and the following operators that map between them: namely (1) *stream-to-relation*, (2) *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; (2) is taken from the relational DB setting; For (3), they defined *insert stream*, *delete stream*, and *relation stream* operators. Further, they developed a method for query execution and benchmarked a prototype implementation on a highway tolling application scenario to show the 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 [19] 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 *query-rewriting-based* approaches like the work of Calbimonte et al. [9] and Özçep et al. [21]. The latter two are of particular interest as these combine OBDA and stream processing.

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

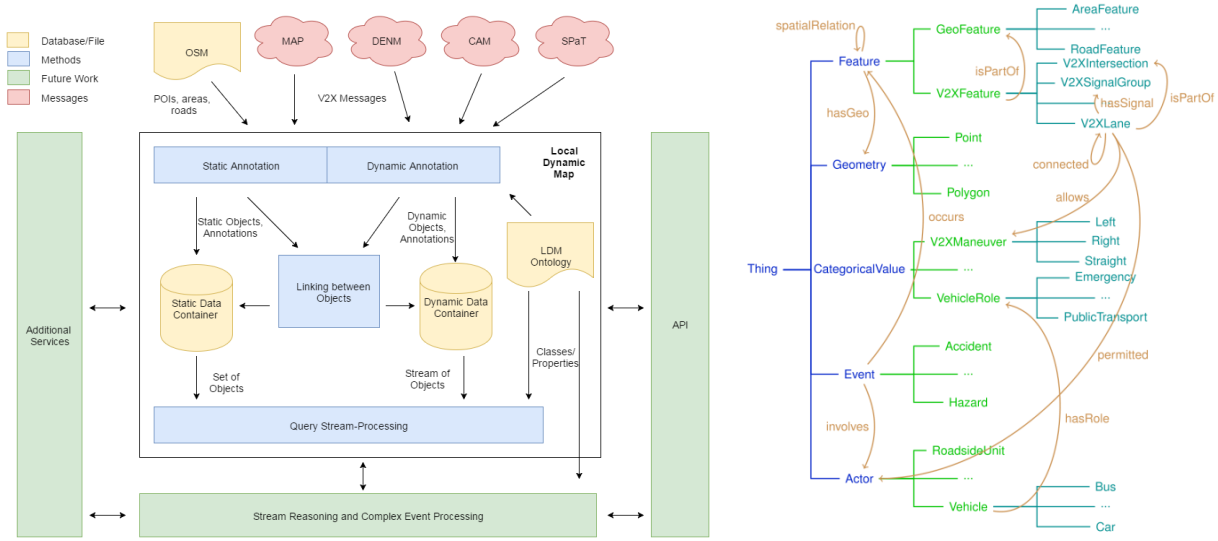

Figure 2 - (a) System architecture and (b) LDM ontology (partial rendering)

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. 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]. Refining the initial components and working out technical details is on-going and future work.

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

- `V2XFeature`: V2X features model the spatial aspect of V2X;
- `GeoFeature` : Geospatial features represent the GIS aspects of the LDM;
- `Geometry` : The geometrical representation of both features;
- `Actor`: The main actors involved in a transport scene;
- `Event`: events which are related to the transport domain;
- Properties for partonomies (`isPartOf`), spatial relations (`intersects`), connectivity (`connected`), and others like `isAllowed`, `hasRole`, and `isManaged`.

We have modeled the ontology to capture all elements of a traffic scene. For instance, geospatial features define POIs and roads. These classes are also linked to GeoOWL and GeoNames for embedding it into existing ontologies.[8] V2X features are the domain specific modeling of the MAP topology and describe the details of an intersection including its lanes, roadside units, allowed maneuvers, traffic lights, etc. The `Actor class`, e.g., person, vehicle, or roadside unit, is linked by the property `permitted` to lanes, etc.

Static and Dynamic Data Container. The Static Data Container (SDC) keeps spatial-relational data that never or infrequently changes, thus storing the first and second layer of the

[7] Available at http://www.kr.tuwien.ac.at/research/projects/loctrafflog/LocalDynamicMapITS-v0.1-Lite.owl
[8] cf. http://www.w3.org/2005/Incubator/geo/XGR-geo/ and http://www.geonames.org/ontology/

LDM, whereas the Dynamic Data Container (DDC) stores the streaming data of the third and fourth layer. The data container is made of the following elements:

- The static DB includes the static ABox (e.g., signal phases) and TBox (e.g., class inclusions);
- The spatial DB is an import of OSM instances (e.g., road network);
- The stream DB uses the capabilities of PipelineDB, i.e., streams and continuous views. The DB is modeled such that there are one-to-one mappings from streams to continuous views, and further to the TBox classes and properties.

The spatial DB contains tables like `Point` or `Polygon`, which store different types of geospatial objects of an OSM map, e.g., the table `Point` includes different POIs like petrol stations; The SDC also contains the table `V2XMAP` that keeps MAP messages, which describe 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). Objects of the first layer are initially added from an OSM instance (e.g., region of Vienna) and stored in the SDC. Objects of the second layer might be predefined for a roadside unit or frequently updated by incoming MAP messages. The DDC is a stream DB based on PipelineDB and contains the tables that store the raw message data (e.g., vehicle positions, signal phases) as streams, which can be accessed using continuous views over the streams.

Annotation and Linking Framework. The Annotation Framework (AF) constantly receives V2X messages, extracts the raw message data, and connects the content to the TBox by adding the data to the SDC and DDC. As we have a one-to-one mapping between classes (e.g., vehicles) resp. properties (e.g., speed) and DB relations, we split the message content into triples, e.g., car1 hasSpeed 50, and add this to the assigned tables, e.g., we add the tuples `<car1>` and `<car1,50>` to the unary table `Vehicle` resp. binary table `hasSpeed`. Each object gets an unique ID assigned, which allows the access of all of its data with a single query.

In the AF, we can take advantage of our work on the city-bench benchmark creation tool [12], 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. The following rules create class and property assertions for police stations in a city, where Geo_{Point} and Tag_{Police} extract the OSM points that are tagged as "police":[9]

$$Geo_{Point}(x) \wedge Tag_{Police}(x) \rightarrow PoliceStation(x)$$

$$Geo_{Point}(x) \wedge Tag_{Police}(x) \wedge Geo_{Polygon}(y) \wedge Tag_{City}(y) \wedge Inside(x, y) \rightarrow hasPoliceStation(y, x)$$

The Linking Framework (LF) computes and stores links between objects that are not directly represented in the data. The linking could be computed off- or online using similar *Datalog* rules as in the AF. For the linking, spatial relations are the main focus, where we follow the standard approach called Dimensionally Extended Nine-Intersection Model (DE-9IM), which supports relations like `Touches`, `Intersects`, and `Disjoint`. One important link is the adjacency of lanes. The following rule calculates which lanes are adjacent to other lanes:

$$Geo_{Polygon}(x) \wedge V2XMAP_{Lane}(x) \wedge Geo_{Polygon}(y) \wedge V2XMAP_{Lane}(y) \wedge intersects(x,y) \rightarrow adjacentLane(x,y)$$

[9] `PoliceStation(x)` and `hasPoliceStation(y, x)` are legible renderings of the N-Triples `x <type> <PoliceStation>.` and `y <hasPoliceStation> x.`

The atom $V2XMAP_{Lane}$ is used to extract incoming MAP messages and allows us to filter different sections of a MAP message, e.g., the lanes. Another important link is the connection of a MAP intersection to its representation in OSM. For this, the OSM representation has to be matched the closest MAP intersection, thus anchoring the V2X features in the OSM road graph.

Query Answering over Streams. This component is central to the usage of a semantically enriched LDM, since the query component allows us to access the data stored in the LDM. For stream query evaluation, there are two approaches possible: *pull-based queries* that evaluate a query at a single point in time (i.e., the query time); *push-based queries* that evaluate a query continuously and return the ongoing changes of the LDM data. We outline our method for pull-based queries and leave the more complex push-based querying for future work. For pull-based queries, we take advantage of our previous work on OBDA for spatial RDBMS [11]. We extend the approach of [11] with a window operator that collects a set of assertions for a query atom from the DDC streams and allow to calculate aggregation functions for numerical (e.g., sum), sequential (e.g., first), and spatial (e.g., line) values. The main algorithm for query evaluation over streams can be sketched as follows:

1. Extract the join tree using hypergraph decomposition from the input query [11], where each node represents either an *ontology atom*, e.g., `MotorVehicle(x)`, *spatial atom*, e.g., `intersects(x,y)`, or *stream atom*, e.g., $position_{(first,30s)}$ `(x,y)`, or the combination of them.
2. Traverse the join tree bottom up, left-to-right to evaluate each set of CQs and keep the intermediate results in-memory. We distinguish between the following nodes:
   (a) *Ontology*: Apply OBDA rewriting [10,22] and evaluate the set of CQ for each atom;
   (b) *Spatial*: Evaluate the spatial relations over the spatial objects and filter accordingly;
   (c) *Stream*: Apply the window operator on the stream for extracting a set of data points and apply grouping and aggregation to the data points.
3. Join the results and project them to the output query variables.

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. Further, the time interval for a window operator and aggregation are defined in the query atom, e.g., $speed_{(avg,30s)}$`(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 [13]. 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. Application Scenarios and Experiments

In this section, we give an overview of three application scenarios that illustrate the usability and benefits of our approach. The scenarios are related to an existing roadside ITS station on a road intersection (called I_1) at Handelskai, Vienna depicted in Figure 3. The ITS station receives any V2X message and is capable to send traffic light signals (SPaT) and its intersection topology (MAP) messages. The first scenario is concerned with static data, where the consistency of the topology is validated offline. The second and third scenario is concerned with dynamic data arising from vehicle movements (CAM) and traffic light signal phases.

Application Scenario 1 (AS1) - Lane Consistency. The focus of this scenario 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. The results of the queries listed below have to be compared to a base query that finds all lanes on an intersection: $q_1(x) = V2XLane(x)\ isPartOf(x,y)\ \wedge V2XIntersection(y)$. Another approach would include negation (i.e., `not`) in queries, but this would lead to an extension of the OBDA methods. The following queries show consistency checks, where `connected(x,y)` is extracted from the MAP messages:

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

   $q_2(x) = V2XLaneInForMotor(x) \wedge connected(x,y) \wedge V2XLaneOutForCars(y) \wedge isPartOf(x,z) \wedge V2XIntersection(z)$

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

   $q_3(x) = V2XLaneIn(x) \wedge connected(x,) \wedge allowsManeuver(x,y) \wedge V2XManeuver(y) \wedge isPartOf(x,y) \wedge V2XIntersection(x)$

The query rewriting of OBDA applies several steps of inference, therefore these queries would be hard to formulate manually in SQL. For instance `V2XLaneInForMotor` is a sub-class of `V2XLaneIn` and the 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 property assertions.

Application Scenario 2 (AS2) - Emergency Detection. The focus of the first dynamic scenario is driven by the aim of detecting emergency vehicles including ambulances and fire trucks on an intersection. For this, the integration of CAM and MAP messages is needed. We use the LDM to detect, whether (a) a vehicle is an emergency vehicle (b) a vehicle is moving on one of the lanes of our intersection, expressed by the following query:

3. Find all emergency vehicles `z` that are currently on our intersection I_1:

   $q_6(z) = V2XIntersection(x) \wedge (x=I_1) \wedge isPartOf(y,x) \wedge V2XLane(y)\ hasGeo(y,u) \wedge intersects(u,v) \wedge position_{(line,\ 10s)}(z,v) \wedge MotorVehicle(z) \wedge hasRole(z,EmergencyRole)$

Since we have dynamic data, the query atom $position_{(line,\ 10s)}(z,v)$ specifies a time-based sliding window of `10s`, where the GPS positions are used to construct a path using the aggregate function `line`; `intersects(u,v)` checks if the path crosses the bounding box of I_1.

Application Scenario 3 (AS3) - Red Light Violation. The focus of this complex dynamic scenario is driven by improving road safety on a heavily frequented intersection enabling the detecting of red light violations (e.g., car crossing at red light). For this, the integration of all the V2X messages is needed. We use the LDM to detect, whether (a) a vehicle is moving above 30kph speed on a lane (b) whose current signal phase will turn red in 4s. This is expressed by the following query:

4. Find all vehicles `y` that are moving above 30kph and violate the signal phase stop of lane `x`:

   $q_7(x,y) = V2XLaneIn(x) \wedge hasGeo(x,u) \wedge intersects(u,v) \wedge position_{(line,\ 4s)}(y,v) \wedge Vehicle(y) \wedge speed_{(avg,\ 4s)}(y,r) \wedge (r>30) \wedge isManaged(x,z) \wedge V2XSignal(z) \wedge hasState_{(first,\ -4s)}(z,Stop)$

The query atoms $position_{(line,\ 4s)}(y,v)$, $speed_{(avg,\ 4s)}(y,r)$, and $hasState_{(first,\ -4s)}(z,Stop)$ specify different time-based sliding windows and different types of aggregation, where `first` gives the first element of a sequence.

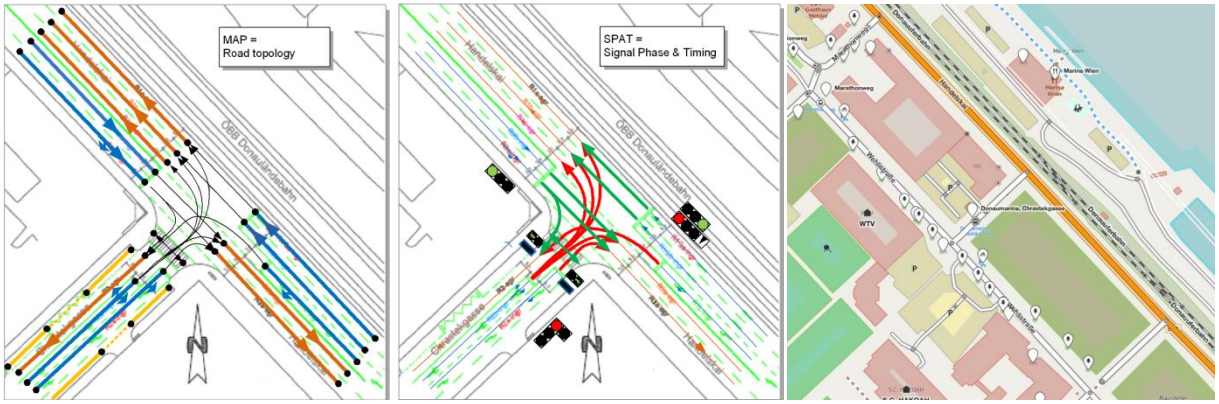

Figure 3 - MAP topology, the related SPaT messages, and OSM of Handelskai

Initial Implementation and Experiments. We give an overview of our initial experimental prototype and a validation of the application scenarios on simulated data for a fixed roadside ITS station. We have extended our prototype of [11] (in Java 1.8) for spatial query answering over streams using PipelineDB 0.9.3 as a stream RDBMS. Our experiments are based on the above scenarios using a simple vehicle simulation with 10 to 5000 vehicles that update their position and speed every second. We conducted our experiments on a Mac OS X 10.6.8 system with an Intel Core i7 2.66GHz and 8 GB of RAM measuring the average of five runs. The experiments show that the static (base case) AS1 needs 1.01s, whereas the streamed AS2 and AS3 need between 1.02s and 2.48s for query evaluation.[10] We observe that already an unoptimized prototype is sufficient to answer complex queries; however, optimizations such as partial evaluation and caching are desired to achieve evaluation times below 1s.

6. Conclusion and Future Work

In this paper, we presented an extension of the LDM with Semantic Web technologies, which allows us to define a world model, i.e., the V2X ontology, an expressive and simple query language, derived model properties, and the capabilities to infer new information over streams. Our envisioned architecture is designed to show how these technologies can be applied in the LDM. The architecture consists of an LDM ontology, an annotation and linking framework, static and dynamic data containers, and a query answering component over streams. We provide the ontology, and show how the architecture can be implemented based on existing work in spatial query answering [11,12]. Then, we developed the scenarios lane consistency, emergency detection, and red light violation to show the usability and benefits of our design.

Future research is directed to extend the components of the framework. In particular, the annotation and linking framework as well as the query component could be elaborated. The latter by realizing push-based querying using query decomposition and caching. Also further investigations regarding expressivity and efficiency are needed. The three application scenarios are good starting points to evaluate the framework and its implementation in a larger test setting including real-life benchmarks using traffic simulations. Finally, future work would be directed towards methods for complex event detection and model-based diagnosis.

[10] Details are available on www.kr.tuwien.ac.at/research/projects/loctrafflog/sr2016/

Acknowledgement

Supported by the Austrian Research Promotion Agency (FFG) project Industrienahe Dissertationen and the Austrian Science Fund (FWF) projects P26471 and P27730.

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, 1431-1438.
8. T. Berners-Lee, J. Hendler, O. Lassila (2001). The semantic web. Scientific American 284(5), 34-43.
9. J. Calbimonte, J. Mora, O. Corcho (2016). Query rewriting in RDF stream processing. In: Proc. of ESWC 2016, LNCS, vol. 9678, Springer, 486–502.
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, 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.
13. 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.
14. ETSI EN 302 895 (V1.1.0) (2014). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM).
15. ISO/TS 17931:2013 (2013). Intelligent transport systems - Extension of map database specifications for Local Dynamic Map for applications of Cooperative ITS.
16. ISO/TS 18750:2015 (2015). Intelligent transport systems - Cooperative systems - Definition of a global concept for Local Dynamic Maps.
17. 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
18. E. Koenders, D. Oort, K. Rozema (2014). An open Local Dynamic Map. In: Proc. of ITS European Congress 2014, ERTICO - ITS Europe.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.