Detecting Mobility Patterns using Spatial Query Answering over Streams

Thomas Eiter[1] Patrik Schneider[1,2]
Josiane Xavier Parreira[2]

(1) Institute of Information Systems, Vienna University of Technology, Austria

(2) Siemens CT, Austria

SR 2017, Wien, 22th of October 2017

Motivation - Cooperative-ITS

- Cooperative-ITS Vision (C-ITS)
  - Health & Safety by monitoring
  - Efficient urban mobility by optimizations
  - Help autonomous cars

V2X Overview [ETSI2010]

Motivation - Cooperative-ITS

- Cooperative-ITS Vision (C-ITS)
  - Health & Safety by monitoring
  - Efficient urban mobility by optimizations
  - Help autonomous cars
- Vehicle-to-X communication (V2X)
  - Traffic participants exchange information as V2X messages
  - Real time, simultaneously, and location based

V2X Overview [ETSI2010]

Motivation - Cooperative-ITS

- Cooperative-ITS Vision (C-ITS)
  - Health & Safety by monitoring
  - Efficient urban mobility by optimizations
  - Help autonomous cars
- Vehicle-to-X communication (V2X)
  - Traffic participants exchange information as V2X messages
  - Real time, simultaneously, and location based
- Goal: Find "traffic patterns" by monitoring V2X messages in a complex and fast changing environment

V2X Overview [ETSI2010]

Motivation - Cooperative-ITS

- Cooperative-ITS Vision (C-ITS)
  - Health & Safety by monitoring
  - Efficient urban mobility by optimizations
  - Help autonomous cars
- Vehicle-to-X communication (V2X)
  - Traffic participants exchange information as V2X messages
  - Real time, simultaneously, and location based
- Goal: Find "traffic patterns" by monitoring V2X messages in a complex and fast changing environment
- Use of a spatial-stream database for V2X messages, where a C-ITS domain ontology is build on top [Netten2013]

V2X Overview [ETSI2010]

Three Scenarios - What are Patterns?

- S1 - Traffic statistics:
  1. Object level
  2. Road/Lane level
  3. Intersection level
  4. Network level
- S2 - Vehicle maneuvers:
  1. Slow down or speed up
  2. Drive straight on, turn left, turn right
  3. Stop, unload, park
  4. Lane change
  5. Overtake, u-turn
- S3 - Event detection:
  1. Red-light violation
  2. Obstructed view
  3. Accident
  4. Traffic rule violation
  5. Traffic congestion

Vissim Traffic Simulation Luxembourg City

Desired Features

Feature	Details
F1: Time model	Point-based or interval-based model
F2: Process model	Push-based, pull-based, or combined queries
F3: Spatial relations	Point-set model, more detailed dim. ext. 9-Intersection model, or qualitative spatial reasoning (e.g. RCC8)
F4: Temporal relations	Linear temporal logic (LTL), Allen's time interval algebra, or Metric temporal logic (MTL)
F5: Numerical aggregations	Aggregations (e.g., sum) on a set or multiset (bag) of data items
F6: Spatial aggregations	Build geometric objects (e.g., paths) from more granular objects (e.g., points)
F7: Numerical predictions	Prediction of new data items (using e.g., linear regression)
F8: Trajectory predictions	Predict (possible) movements of vehicle (e.g., linear path)
F9: Geo matching	Match to geometric object
F10: Advanced	Graph connectivity, negation as failure, and repairs

Requirements Matrix

Case	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
S1.1	Po	Pull	N	N	Y	P	P	P	N	
S1.2	Po	Pull	Y	N	Y	Y	P	P	N	
S1.3	Po	Pull	Y	N	Y	Y	P	P	P	
S1.4	Int	Pull	Y	Y	Y	Y	Y	Y	P	CN
S2.1	Po	Pull	N	N	Y	P	P	N	N	
S2.2	Po	Pull	Y	N	Y	Y	P	Y	Y	
S2.3	Po	Pull	Y	P	Y	Y	Y	N	N	
S2.4	Po	Push	Y	N	Y	Y	P	P	P	
S2.5	Bo	Push	Y	N	Y	Y	Y	Y	P	
S3.1	Bo	Comb	Y	P	Y	Y	Y	Y	Y	
S3.2	Bo	Comb	Y	P	Y	Y	Y	Y	Y	
S3.3	Int	Comb	Y	Y	Y	Y	P	P	Y	NA
S3.4	Int	Comb	Y	Y	Y	Y	Y	Y	Y	NA, CN
S3.5	Int	Comb	Y	Y	Y	Y	Y	Y	Y	NA, CN

- Legend:
  - Y: required, N: not required, P: possibly required
  - Po: point-based; Int: interval-based; Bo: Both possible
  - Push: push-based; Pull: pull-based; Comb: combined
  - NA: Negation as Failure; CN: graph connectivity

Streams and Pulses

- Data model: point-based (vs. interval-based) and valid time (vs. transaction time)
- A data stream is a triple $F = (\mathbb{T}, v, P)$:
  - Timeline $\mathbb{T}$, which is a *closed* interval of $(\mathbb{N}, \leq)$
  - Function $v : \mathbb{T} \rightarrow \mathcal{F}$ that assigns to each element of $\mathbb{T}$, data items (one ABox assertions) of a stream database $\mathcal{S}_{\mathcal{F}}$
  - Pulse P is the general interval of consecutive data items [Özçep2015]

Streams and Pulses

- Data model: point-based (vs. interval-based) and valid time (vs. transaction time)
- A data stream is a triple $F = (\mathbb{T}, v, P)$:
  - Timeline $\mathbb{T}$, which is a *closed* interval of $(\mathbb{N}, \leq)$
  - Function $v : \mathbb{T} \rightarrow \mathcal{F}$ that assigns to each element of $\mathbb{T}$, data items (one ABox assertions) of a stream database $\mathcal{S}_{\mathcal{F}}$
  - Pulse P is the general interval of consecutive data items [Özçep2015]
- Example 1: Timeline $\mathbb{T} = [0, 10]$ with two streams:

Stream of vehicles $F_1 = (\mathbb{T}, v, 1)$:
$v(0) = \{speed(c_1, 30),\ pos(c_1, (5, 5)),\ speed(b_1, 10),\ pos(b_1, (1, 1))\}$,
$v(1) = \{speed(c_1, 29),\ pos(c_1, (6, 5)),\ speed(b_1, 5),\ pos(b_1, (2, 1))\}$
$v(2) = \{speed(c_1, 34),\ pos(c_1, (7, 5))\}$

Stream of signal phases $F_2 = (\mathbb{T}, v, 3)$:
$v(0) = \{hasState(t_1, Green)\}, v(3) = \{hasState(t_1, Red)\}, v(6) = \{hasState(t_1, Green)\}$

Spatial-stream Conjunctive Queries

- Extend CQ with spatial and stream atoms over a DL-Lite$_A$ KB
  - CQ have answer **x** resp. existentially quantified **y** variables

$$q(x,y): \quad LaneIn(x) \wedge hasLocation(x,u) \wedge intersects(u,v) \wedge pos_{(line,\ 4s)}(y,v) \wedge Vehicle(y) \wedge speed_{(avg,\ 4s)}(y,r) \wedge (r > 30) \wedge isManaged(x,z) \wedge SignalGroup(z) \wedge hasState_{(first,\ -4s)}(z, Stop)$$

- We have our three types of atoms:
  - $Q_{O_i}(\mathbf{x},\mathbf{y})$: Q_{O_i} is a concept/role atom, unfold regarding $\mathcal{T}$ of KB
  - $Q_{S_j}(\mathbf{x},\mathbf{y})$: Q_{S_j} is a spatial relation or a localization
  - $Q_{F_k}(\mathbf{x},\mathbf{y})$: Q_{F_k} is a stream atom

Spatial-stream Conjunctive Queries

- Extend CQ with spatial and stream atoms over a DL-Lite$_A$ KB
  - CQ have answer **x** resp. existentially quantified **y** variables

$$q(x,y): \; LaneIn(x) \wedge hasLocation(x,u) \wedge intersects(u,v) \wedge pos_{(line,\ 4s)}(y,v) \wedge Vehicle(y) \wedge speed_{(avg,\ 4s)}(y,r) \wedge (r > 30) \wedge isManaged(x,z) \wedge SignalGroup(z) \wedge hasState_{(first,\ -4s)}(z, Stop)$$

- We have our three types of atoms:
  - $Q_{O_i}(\mathbf{x},\mathbf{y})$: Q_{O_i} is a concept/role atom, unfold regarding $\mathcal{T}$ of KB
  - $Q_{S_j}(\mathbf{x},\mathbf{y})$: Q_{S_j} is a spatial relation or a localization
  - $Q_{F_k}(\mathbf{x},\mathbf{y})$: Q_{F_k} is a stream atom
- A closer look at stream atoms Q_{F_k}:
  - $Q_{F_k}\ (agr, L)$: aggregate of last/next L time units (relative to query time)
  - $Q_{F_k}\ (agr, O)$: aggregate of all previous L time units
  - $Q_{F_k}\ (agr, L, T)$: tuples that are between L and T (historic data)

Spatial-stream Conjunctive Queries

- Extend CQ with spatial and stream atoms over a DL-Lite$_A$ KB
  - CQ have answer **x** resp. existentially quantified **y** variables

$$q(x,y): \; LaneIn(x) \wedge hasLocation(x,u) \wedge intersects(u,v) \wedge pos_{(line,\ 4s)}(y,v) \wedge Vehicle(y) \wedge speed_{(avg,\ 4s)}(y,r) \wedge (r > 30) \wedge isManaged(x,z) \wedge SignalGroup(z) \wedge hasState_{(first,\ -4s)}(z, Stop)$$

- We have our three types of atoms:
  - $Q_{O_i}(\mathbf{x},\mathbf{y})$: Q_{O_i} is a concept/role atom, unfold regarding $\mathcal{T}$ of KB
  - $Q_{S_j}(\mathbf{x},\mathbf{y})$: Q_{S_j} is a spatial relation or a localization
  - $Q_{F_k}(\mathbf{x},\mathbf{y})$: Q_{F_k} is a stream atom
- A closer look at stream atoms Q_{F_k}:
  - $Q_{F_k}(agr, L)$: aggregate of last/next L time units (relative to query time)
  - $Q_{F_k}(agr, O)$: aggregate of all previous L time units
  - $Q_{F_k}(agr, L, T)$: tuples that are between L and T (historic data)
- Aggregate function *agr* can be:
  - Numerical: *count*, *min*, *max*, *sum*, *mean*, *sd*
  - Prediction: *linreg*, *loglinreg*, *polyreg* (simple models)
  - Position: *first*, *last*
  - Spatial: *point*, *line*, *angle*, *tree*, *area*, *traject*

Query Answering by Stream Aggregation

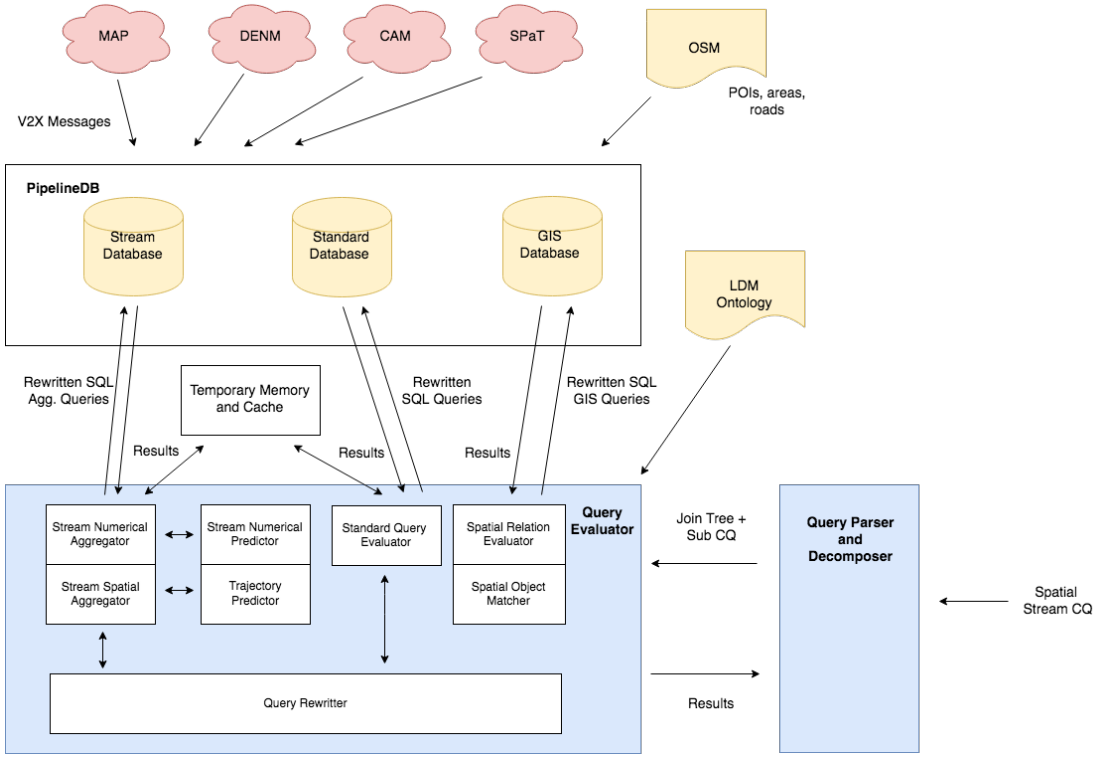
- Goal: Pull-based spatial-stream CQ in ontology-mediated QA
- Challenges:
  - How to untangle different types of query atoms?
  - Clear semantics for QA?
  - Evaluation on an RDBMS (detect red-light violations below 1s):
    - LOGSPACE data complexity
  - Problems with aggregates in DL-Lite$_A$:
    - Certain answers semantics $\rightarrow$ Intersection of answers over all possible models of the KB $\rightarrow$ Empty models

Query Answering by Stream Aggregation

- Goal: Pull-based spatial-stream CQ in ontology-mediated QA
- Challenges:
  - How to untangle different types of query atoms?
  - Clear semantics for QA?
  - Evaluation on an RDBMS (detect red-light violations below 1s):
    - LOGSPACE data complexity
  - Problems with aggregates in DL-Lite$_A$:
    - Certain answers semantics $\rightarrow$ Intersection of answers over all possible models of the KB $\rightarrow$ Empty models
- Solution:
  - Staging (in-memory): (1) Stream detemporalization, (2) Standard rewriting, (3) Spatial evaluation
    $\rightarrow$ Hypertree decomposition [Maier1983]
  - Stream Aggregation *by detemporalizing* stream atoms
    $\rightarrow$ Epistemic Aggregate Queries (EAQ) [Calvanese2008]

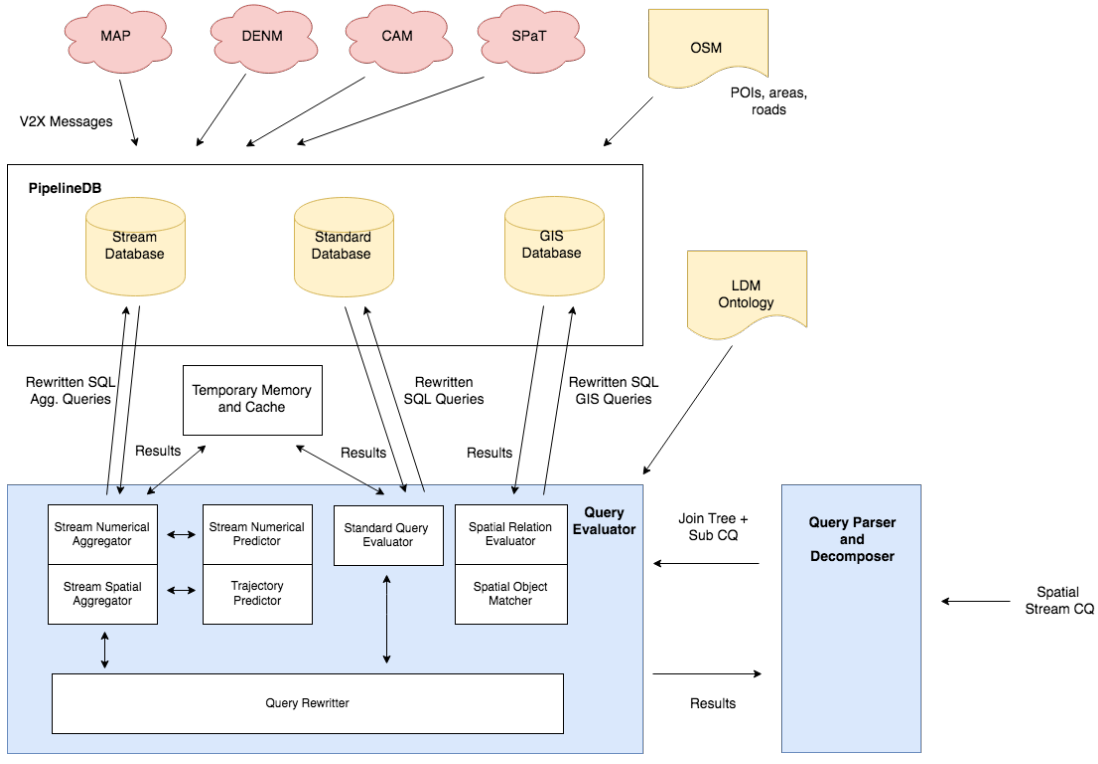
Stream Query Platform

- System architecture:



Stream Query Platform

- System architecture:



- Components:
  - Spatial-stream RDBMS: PipelineDB
  - Query parser and decomposer: hypertree decomposition (preprocessing)
  - Ontology evaluator: Owlgres 0.1 [Stocker2008] rewriting
  - Stream evaluator: stream detemporalization by grouping and aggregation
  - Spatial evaluator: evaluated using JTS Topology Suite

Finished / Open Features

- Finished:
  - F3 - Spatial relations: Point-set model done, RCC8 nice extension
  - F5 - Numerical aggregations: Fully done
  - F6 - Spatial aggregations: Implemented, specific aggregates (e.g., convex hull) missing

Finished / Open Features

- Finished:
  - F3 - Spatial relations: Point-set model done, RCC8 nice extension
  - F5 - Numerical aggregations: Fully done
  - F6 - Spatial aggregations: Implemented, specific aggregates (e.g., convex hull) missing
- Partially done:
  - F1 - Time model: Point-based model done, but interval-based model needed
  - F2 - Process model: Pull-based queries done, push-based desired, but tricky with PipelineDB
  - F7 - Numerical predictions: Linear regression done, other methods challenging, i.e., model building on top of streams
  - F8 - Trajectories: Linear path extension done, but map matching (to a road graph) missing

Finished / Open Features

- Finished:
  - F3 - Spatial relations: Point-set model done, RCC8 nice extension
  - F5 - Numerical aggregations: Fully done
  - F6 - Spatial aggregations: Implemented, specific aggregates (e.g., convex hull) missing
- Partially done:
  - F1 - Time model: Point-based model done, but interval-based model needed
  - F2 - Process model: Pull-based queries done, push-based desired, but tricky with PipelineDB
  - F7 - Numerical predictions: Linear regression done, other methods challenging, i.e., model building on top of streams
  - F8 - Trajectories: Linear path extension done, but map matching (to a road graph) missing
- Open:
  - F4 - Temporal relations: Not included yet (important for Scenario 3), LTL [Thost2015], MTL [Brandt2017], Allen's Time Interval Algebra extension for DL-Lite$_A$
  - F9/10 - Advanced: Features (e.g., transitivity) go beyond DL-Lite$_A$ and FO-rewritability, different language needed

Challenges

- Syntax: Detect red-light violations by vehicles that might speed above 30km/h

$$q_1(x,y): \quad LaneIn(x) \wedge hasLocation(x,u) \wedge intersects(u,v) \wedge pos(y,v)[traject,-10] \wedge Vehicle(y) \wedge speed(y,r)[linreg,avg,-10] \wedge (r>30) \wedge isManaged(x,z) \wedge SignalGroup(z) \wedge hasState(z,Stop)[first,-5]$$

- Sub queries: Count vehicles that start at I_1 and pass through either I_2 or I_3

$$q_a(y,u): \quad Vehicle(y) \wedge pos(y,z,u)[line,240] \wedge intersects(y,u) \wedge hasGeo(x,u) \wedge Intersection(x) \wedge (x=I_1)$$
$$q_b(y): \quad Vehicle(y) \wedge pos(y,z,v)[line,60] \wedge intersects(z,u) \wedge hasGeo(x,u) \wedge Intersection(x) \wedge (x=I_2) \wedge before(u,v) \wedge q_a(w,u) \wedge (y=w)$$
$$q_c(y): \quad Vehicle(y) \wedge pos(y,z,v)[line,60] \wedge intersects(y,u) \wedge hasGeo(x,u) \wedge Intersection(x) \wedge (x=I_3) \wedge before(u,v) \wedge q_a(w,u) \wedge (y=w)$$
$$q_2(x,y): \quad q_b(x)[count,60] \wedge q_c(y)[count,60]$$

- Matching: Count vehicles that might turn left/right or head-straight on

$$q_a(x,y): \quad Vehicle(y) \wedge pos(y,z)[traject,-10] \wedge match(z,Left) \wedge intersects(z,u) \wedge hasGeo(x,u) \wedge Intersection(x)$$
$$q_b(x,y): \quad Vehicle(y) \wedge pos(y,z)[traject,-10] \wedge match(z,Straight) \wedge intersects(z,u) \wedge hasGeo(x,u) \wedge Intersection(x)$$
$$q_c(x,y): \quad Vehicle(y) \wedge pos(y,z)[traject,-10] \wedge match(z,Right) \wedge intersects(z,u) \wedge hasGeo(x,u) \wedge Intersection(x)$$
$$q_3(z,u,v,w): \quad q_a(z,u)[avg,60] \wedge q_b(z,v)[avg,60] \wedge q_c(z,w)[avg,60]$$

- Order, long-term memory: Detect cars that perform an u-turn

$$q_a(x,z): \quad Car(y) \wedge pos(x,y,z)[line,30,15] \wedge match(z,Straight)$$
$$q_b(x,z): \quad Car(y) \wedge pos(x,y,z)[line,15,10] \wedge match(z,Left)$$
$$q_c(x,z): \quad Car(y) \wedge pos(x,y,z)[line,10,5] \wedge match(z,Left)$$
$$q_d(x,z): \quad Car(y) \wedge pos(x,y,z)[line,5] \wedge match(z,Straight)$$
$$q_4(x): \quad q_a(x,u) \wedge before(u,v) \wedge q_b(x,v) \wedge before(v,w) \wedge q_c(x,w) \wedge before(w,t) \wedge q_d(x,t)$$

Conclusion

- Aim to extend spatial QA over streams to detect mobility patterns
- Future work (short-term):
  - Push-based queries → Allow a combination with pull-based queries
  - Long-term memory for patterns extracted from streams
  - Spatial matching that includes map matching for trajectories
  - Optimizations → Improved caching and "better" query rewriting
  - Bag semantics → inconsistencies in larger windows → Repairs
- Future work (long-term):
  - Interval-based data model →Temporal relations (Allen Time Interval Algebra)
  - Allow (spatial topological relations (RCC8) → Datalog-rewriting [Koubarakis2017]
  - Statistical model building on streams for predictions
  - Lift to SPARQL

References I

1 Andreone, L., Brignolo, R., Damiani, S., Sommariva, F., Vivo, G., Marco, S.: Safespot final report. Tech. Rep. D8.1.1 (2010), available online.

2 Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)

3 Calvanese, D., Kharlamov, E., Nutt, W., Thorne, C.: Aggregate queries over ontologies. In: Proc. of ONISW 2008. pp. 97–104 (2008)

4 Eiter, T., Krennwallner, T., and Schneider, P.: Lightweight spatial conjunctive query answering using keywords. In Proc. of ESWC 2013, pages 243258, 2013.

5 Maier, D.: The Theory of Relational Databases. Computer Science Press (1983)

6 Netten, B., Kester, L., Wedemeijer, H., Passchier, I., Driessen, B.: Dynamap: A dynamic map for road side its stations. In: Proc. of ITS World Congress 2013 (2013)

7 Özçep, Ö.L., Möller, R., Neuenstadt, C.: Stream-query compilation with ontologies. In: Proc. of AI 2015. pp. 457–463 (2015)

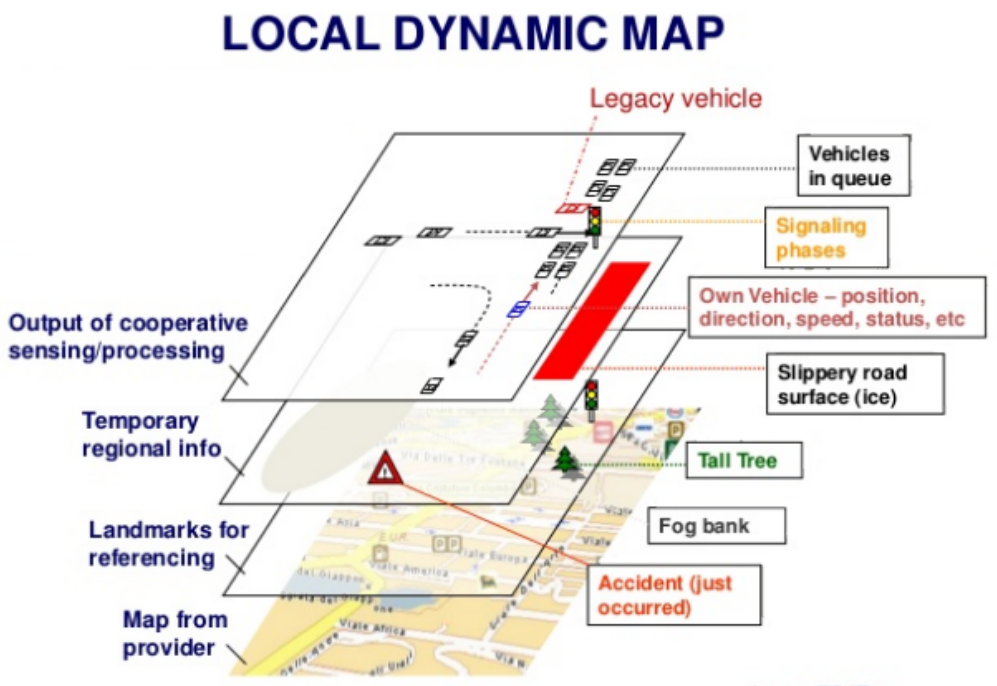
8 Stocker, M., Smith, M.: Owlgres: A scalable OWL Reasoner. In: Proc. of OWLED2008 (2008)

Local Dynamic Map

- LDM is a integration platform for V2X messages with GIS maps [Safespot2010]

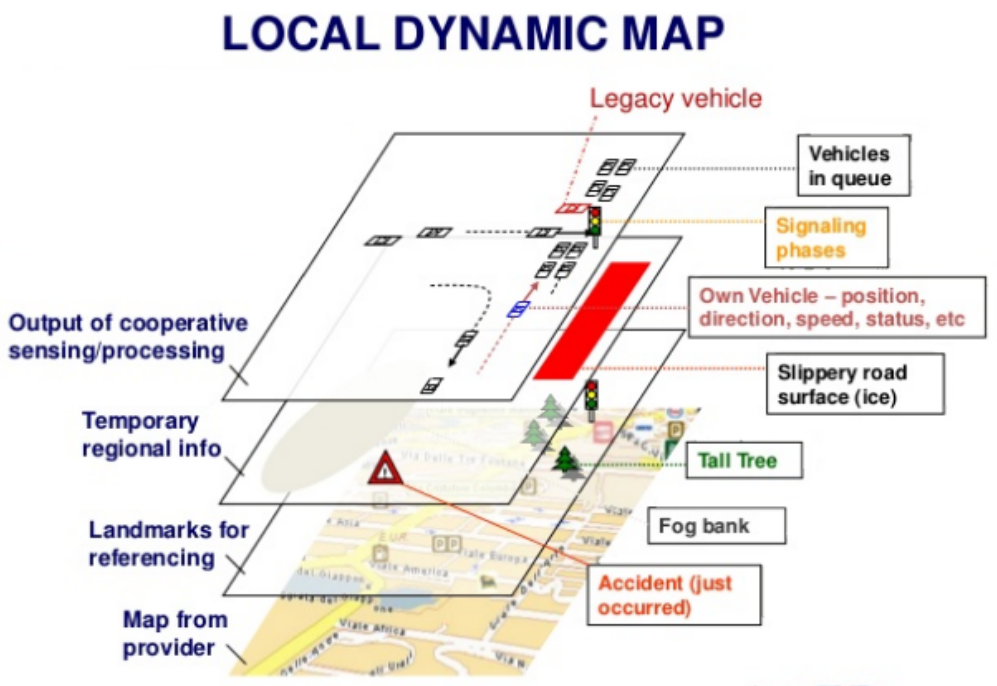
Local Dynamic Map

- LDM is a integration platform for V2X messages with GIS maps [Safespot2010]
- Has four layers of information:
  - Highly dynamic (e.g., sensing)
  - Temporary regional (e.g., weather, signal phases)
  - Transient static (e.g., topology)
  - Static (e.g., GIS maps)



Local Dynamic Map

- LDM is a integration platform for V2X messages with GIS maps [Safespot2010]
- Has four layers of information:
  - Highly dynamic (e.g., sensing)
  - Temporary regional (e.g., weather, signal phases)
  - Transient static (e.g., topology)
  - Static (e.g., GIS maps)



- Represented by a spatial-stream database for V2X messages, where a C-ITS domain ontology is build on top [Netten2013]

Semantics of DL-Lite$_A$ (S,F)

- Semantics for spatial extension: given in [Eiter2013]
- Semantics for streaming: Interpret the stream over the full $\mathbb{T}$, point-based model
- We define as a sequence $\mathcal{I}_F = (\mathcal{I}_i)_{\mathbb{T}_{min} \leq i \leq \mathbb{T}_{max}}$ of interpretations $\mathcal{I}_i = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}_i} \rangle$
- Then $\mathcal{I}_F$ is a model of $\mathcal{K}_{\mathcal{SF}}$, such that $\mathcal{I}_F \models \mathcal{K}_{\mathcal{SF}}$ iff $\mathcal{I}_i \models \mathcal{S}_{\mathcal{F}i}$ and $\mathcal{I}_i \models \mathcal{A}$, and $\mathcal{I}_i \models \mathcal{T}$, for all $i \in \mathbb{T}$.
- The stream axioms $(stream_F\ C)$ and $(stream_F\ R)$ are interpreted along the same line:

$$(stream_F\ C)^{\mathcal{I}} = \bigcap_{i \in \mathbb{T}} (\{e \in \Delta^{\mathcal{I}} \mid e \in C^{\mathcal{I}_i}\})$$
$$(stream_F\ R)^{\mathcal{I}} = \bigcap_{i \in \mathbb{T}} (\{(a_1, a_2) \mid (a_1, a_2) \in R^{\mathcal{I}_i}\})$$

- Only for satisfiability of the KB, no QA yet!

QA by Stream Aggregation II

- Example 2: EAQ with query time $\mathbb{T}_3$

  $speed_{(avg,\ 4s)}(y, r)$ $pos_{(line,\ 4s)}(y, v)$ $hasState_{(first,\ -4s)}(z, m)$

(1) Windowed ABoxes:
$\mathcal{A}_{\boxplus[0,3]} = \mathcal{A} \bigcup_{0 \leq i \leq 3} \mathcal{A}_i$ and $\mathcal{A}_{\boxplus[3,7]} = \mathcal{A} \bigcup_{3 \leq i \leq 7} \mathcal{A}_i$

(2) Grouping:
q_{speed}: $G_{c_1} = \{|29, 30, 34|\}$ and $G_{b_1} = \{|10, 5|\}$ (or $G_{b_1} = \{|10, 5, 5|\}$?)
q_{pos}: $G_{c_1} = \{|(5, 5), (6, 5), (7, 5)|\}$ and $G_{b_1} = \{|(1, 1), (2, 1)|\}$
$q_{hasState}$: $G_{t_1} = \{|Red, Green|\}$

(3) Aggregation:
$avg(q_{speed}) = \{(c_1, 31), (b_1, 7.5), (l_1, 0)\}$
$line(q_{pos}) = \{(c_1, ((5, 5), (6, 5), (7, 5))), (b_1, ((1, 1), (2, 1)))\}$
$first(q_{hasState}) = \{(t_1, Red)\}$

- Two semantics for data items validity!
- Detemporalize stream atoms:

(1) Create windowed ABoxes:
$\mathcal{A}_{\boxplus_k} = \mathcal{A} \cup \bigcup \{\mathcal{A}_i \mid w_s \leq i \leq w_e\}$, w_s resp. w_e is from L and $\mathbb{T}_i$

(2) Drop temporal order in $\mathcal{A}_{\boxplus_k}$ $\rightarrow$ Bags (multisets) of data items $\rightarrow$ Grouping of bags

(3) Eval aggregate functions on bags