Detecting Mobility Patterns using Spatial Query Answering over Streams

Thomas Eiter\textsuperscript{1}  \hspace{1cm} Patrik Schneider\textsuperscript{1,2}  \hspace{1cm} Josiane Xavier Parreira\textsuperscript{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]
Introduction

Scenarios, Features, and Requirements

Overview Methods and Technology

Qualitative Evaluation

Conclusion and Future Work
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

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

<table>
<thead>
<tr>
<th>Case</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
<th>F10</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1.1</td>
<td>Po</td>
<td>Pull</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>S1.2</td>
<td>Po</td>
<td>Pull</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>S1.3</td>
<td>Po</td>
<td>Pull</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>S1.4</td>
<td>Int</td>
<td>Pull</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td></td>
<td>CN</td>
</tr>
<tr>
<td>S2.1</td>
<td>Po</td>
<td>Pull</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>S2.2</td>
<td>Po</td>
<td>Pull</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>S2.3</td>
<td>Po</td>
<td>Pull</td>
<td>Y</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>S2.4</td>
<td>Po</td>
<td>Push</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>S2.5</td>
<td>Bo</td>
<td>Push</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3.1</td>
<td>Bo</td>
<td>Comb</td>
<td>Y</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>S3.2</td>
<td>Bo</td>
<td>Comb</td>
<td>Y</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>S3.3</td>
<td>Int</td>
<td>Comb</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>NA</td>
</tr>
<tr>
<td>S3.4</td>
<td>Int</td>
<td>Comb</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA, CN</td>
</tr>
<tr>
<td>S3.5</td>
<td>Int</td>
<td>Comb</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>NA, CN</td>
</tr>
</tbody>
</table>

### 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
Introduction

Scenarios, Features, and Requirements

Overview Methods and Technology

Qualitative Evaluation

Conclusion and Future Work
Streams and Pulses

- **Data model**: point-based (vs. interval-based) and valid time (vs. transaction time)

- A **data stream** is a triple \( F = (\mathbb{T}, \nu, P) \):
  - **Timeline** \( \mathbb{T} \), which is a *closed* interval of \((\mathbb{N}, \leq)\)
  - **Function** \( \nu : \mathbb{T} \to \mathcal{F} \) that assigns to each element of \( \mathbb{T} \), data items (one ABox assertions) of a stream database \( S_\mathcal{F} \)
  - **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 \( F = (T, v, P) \):
  - **Timeline** \( T \), which is a *closed* interval of \((\mathbb{N}, \leq)\)
  - **Function** \( v : T \rightarrow F \) that assigns to each element of \( T \), data items (one ABox assertions) of a stream database \( S_F \)
  - **Pulse** \( P \) is the general interval of consecutive data items [Özçep2015]
- **Example 1**: Timeline \( T = [0, 10] \) with two streams:

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

  Stream of signal phases \( F_2 = (T, v, 3) \):
  - \( v(0) = \{\text{hasState}(t_1, \text{Green})\} \), \( v(3) = \{\text{hasState}(t_1, \text{Red})\} \), \( v(6) = \{\text{hasState}(t_1, \text{Green})\} \)
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
    \[
    q(x, y) : \text{LaneIn}(x) \land \text{hasLocation}(x, u) \land \text{intersects}(u, v) \land \text{pos}(\text{line}, 4s)(y, v)
    \land \text{Vehicle}(y) \land \text{speed}_{\text{avg}, 4s}(y, r) \land (r > 30) \land \text{isManaged}(x, z)
    \land \text{SignalGroup}(z) \land \text{hasState}_{\text{first}, -4s}(z, \text{Stop})
    \]
- We have our three types of atoms:
  - \( Q_{O_i}(x, y) \): \( Q_{O_i} \) is a concept/role atom, unfold regarding \( T \) of KB
  - \( Q_{S_j}(x, y) \): \( Q_{S_j} \) is a spatial relation or a localization
  - \( Q_{F_k}(x, y) \): \( Q_{F_k} \) 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
    $$q(x, y) : \text{LaneIn}(x) \land \text{hasLocation}(x, u) \land \text{intersects}(u, v) \land \text{pos}(\text{line}, 4s)(y, v)$$
    $$\land \text{Vehicle}(y) \land \text{speed}_{\text{avg}, 4s}(y, r) \land (r > 30) \land \text{isManaged}(x, z)$$
    $$\land \text{SignalGroup}(z) \land \text{hasState}_{\text{first}, -4s}(z, \text{Stop})$$

- We have our three types of atoms:
  - $Q_{O_i}(x, y)$: $Q_{O_i}$ is a concept/role atom, unfold regarding $T$ of KB
  - $Q_{S_j}(x, y)$: $Q_{S_j}$ is a spatial relation or a localization
  - $Q_{F_k}(x, y)$: $Q_{F_k}$ 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
  \[ q(x, y) : \quad \text{LaneIn}(x) \land \text{hasLocation}(x, u) \land \text{intersects}(u, v) \land \text{pos}_{\text{line}}(y, v) \land \text{Vehicle}(y) \land \text{speed}_{\text{avg}, 4s}(y, r) \land (r > 30) \land \text{isManaged}(x, z) \land \text{SignalGroup}(z) \land \text{hasState}_{\text{first}, -4s}(z, \text{Stop}) \]

- We have our three types of atoms:
  - \( Q_{O_i}(x, y) \): \( Q_{O_i} \) is a concept/role atom, unfold regarding \( T \) of KB
  - \( Q_{S_j}(x, y) \): \( Q_{S_j} \) is a spatial relation or a localization
  - \( Q_{F_k}(x, y) \): \( Q_{F_k} \) 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: \( \text{count}, \text{min}, \text{max}, \text{sum}, \text{mean}, \text{sd} \)
  - Prediction: \( \text{linreg}, \text{loglinreg}, \text{polyreg} \) (simple models)
  - Position: \( \text{first}, \text{last} \)
  - Spatial: \( \text{point}, \text{line}, \text{angle}, \text{tree}, \text{area}, \text{traject} \)
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):
    - \( \text{LOGSPACE} \) data complexity
  - Problems with aggregates in \( \text{DL-Lite}_A \):
    - 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 \( \text{by detemporalizing} \) stream atoms
    \( \rightarrow \) Epistemic Aggregate Queries (EAQ) [Calvanese2008]
Stream Query Platform

- System architecture:

  - PipelineDB
    - Stream Database
    - Standard Database
    - GIS Database

  - Temporal Memory and Cache
  - Query Rewriter

  - Spatial Aggregator
  - Stream Numerical Aggregator
  - Query Tree + Sub CQ
  - Join Tree + Sub CQ

- 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

- Inputs:
  - V2X Messages
  - MAP
  - DENM
  - CAM
  - SPaT

- Outputs:
  - PDIs, areas, roads
Stream Query Platform

- **System architecture:**

![Diagram of 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
Introduction

Scenarios, Features, and Requirements

Overview Methods and Technology

Qualitative Evaluation

Conclusion and Future Work
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
  - F9/10 - Advanced: Features (e.g., transitivity) go beyond DL-Lite A and FO-rewritability, different language needed
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
  - F9/10 - Advanced: Features (e.g., transitivity) go beyond DL-Lite and FO-rewritability, different language needed
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 PIPELINE DB
  - **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

  \[ q_1(x, y) : \ Laneln(x) \land hasLocation(x, u) \land intersects(u, v) \land pos(y, v)[\text{traject}, -10] \land Vehicle(y) \land speed(y, r)[\text{linreg}, \text{avg}, -10] \land (r > 30) \land isManaged(x, z) \land SignalGroup(z) \land hasState(z, \text{Stop})[\text{first}, -5] \]

- **Sub queries:** Count vehicles that start at \( l_1 \) and pass through either \( l_2 \) or \( l_3 \)

  \[ q_a(y, u) : \ Vehicle(y) \land pos(y, z, u)[\text{line}, 240] \land intersects(y, u) \land hasGeo(x, u) \land Intersection(x) \land (x = l_1) \]

  \[ q_b(y) : \ Vehicle(y) \land pos(y, z, v)[\text{line}, 60] \land intersects(z, u) \land hasGeo(x, u) \land Intersection(x) \land (x = l_2) \land before(u, v) \land q_a(w, u) \land (y = w) \]

  \[ q_c(y) : \ Vehicle(y) \land pos(y, z, v)[\text{line}, 60] \land intersects(y, u) \land hasGeo(x, u) \land Intersection(x) \land (x = l_3) \land before(u, v) \land q_a(w, u) \land (y = w) \]

  \[ q_2(x, y) : \ q_b(x)[\text{count}, 60] \land q_c(y)[\text{count}, 60] \]

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

  \[ q_a(x, y) : \ Vehicle(y) \land pos(y, z)[\text{traject}, -10] \land match(z, \text{Left}) \land intersects(z, u) \land hasGeo(x, u) \land Intersection(x) \]

  \[ q_b(x, y) : \ Vehicle(y) \land pos(y, z)[\text{traject}, -10] \land match(z, \text{Straight}) \land intersects(z, u) \land hasGeo(x, u) \land Intersection(x) \]

  \[ q_c(x, y) : \ Vehicle(y) \land pos(y, z)[\text{traject}, -10] \land match(z, \text{Right}) \land intersects(z, u) \land hasGeo(x, u) \land Intersection(x) \]

  \[ q_3(z, u, v, w) : \ q_a(z, u)[\text{avg}, 60] \land q_b(z, v)[\text{avg}, 60] \land q_c(z, w)[\text{avg}, 60] \]

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

  \[ q_a(x, z) : \ Car(y) \land pos(x, y, z)[\text{line}, 30, 15] \land match(z, \text{Straight}) \]

  \[ q_b(x, z) : \ Car(y) \land pos(x, y, z)[\text{line}, 15, 10] \land match(z, \text{Left}) \]

  \[ q_c(x, z) : \ Car(y) \land pos(x, y, z)[\text{line}, 10, 5] \land match(z, \text{Left}) \]

  \[ q_d(x, z) : \ Car(y) \land pos(x, y, z)[\text{line}, 5] \land match(z, \text{Straight}) \]

  \[ q_4(x) : \ q_a(x, u) \land before(u, v) \land q_b(x, v) \land before(v, w) \land q_c(x, w) \land before(w, t) \land q_d(x, t) \]
Introduction

Scenarios, Features, and Requirements

Overview Methods and Technology

Qualitative Evaluation

Conclusion and Future Work
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


Local Dynamic Map

- LDM is an integration platform for V2X messages with GIS maps [Safespot2010]
Local Dynamic Map

- LDM is an 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 an 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 built 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
  \[\mathcal{I}_F = (\mathcal{I}_i)_{\mathbb{T}_{\text{min}} \leq i \leq \mathbb{T}_{\text{max}}}\]
  of interpretations \(\mathcal{I}_i = \langle \Delta^\mathcal{I}_i, \mathcal{I}_i \rangle\)
- Then \(\mathcal{I}_F\) is a model of \(\mathcal{K}_{S,F}\), such that
  \(\mathcal{I}_F \models \mathcal{K}_{S,F}\) iff \(\mathcal{I}_i \models S_F\) and \(\mathcal{I}_i \models A\), and \(\mathcal{I}_i \models T\), for all \(i \in \mathbb{T}\).
- The **stream axioms** \(\text{(stream}_F \ C)\) and \(\text{(stream}_F \ R)\) are interpreted along the same line:
  \[
  \text{(stream}_F \ C)^\mathcal{I} = \bigcap_{i \in \mathbb{T}} \{e \in \Delta^\mathcal{I}_i \mid e \in C^{\mathcal{I}_i}\}
  \]
  \[
  \text{(stream}_F \ R)^\mathcal{I} = \bigcap_{i \in \mathbb{T}} \{(a_1, a_2) \mid (a_1, a_2) \in R^{\mathcal{I}_i}\}
  \]
- Only for **satisfiability** of the KB, no QA yet!
Example 2: EAQ with query time $T_3$

\[
\text{speed}_{(\text{avg}, \text{4s})}(y, r) \quad \text{pos}_{(\text{line}, \text{4s})}(y, v) \quad \text{hasState}_{(\text{first}, -\text{4s})}(z, m)
\]

(1) Windowed ABoxes:
\[A_{\oplus[0,3]} = A \cup \bigcup_{0 \leq i \leq 3} A_i \text{ and } A_{\oplus[3,7]} = A \cup \bigcup_{3 \leq i \leq 7} A_i\]

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

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

Two semantics for data items validity!

Detemporalize stream atoms:
(1) Create windowed ABoxes:
\[A_{\oplus_k} = A \cup \bigcup \{A_i \mid w_s \leq i \leq w_e\}, w_s \text{ resp. } w_e \text{ is from } L \text{ and } T_i\]
(2) Drop temporal order in $A_{\oplus_k} \rightarrow$ Bags (multisets) of data items $\rightarrow$ Grouping of bags
(3) Eval aggregate functions on bags