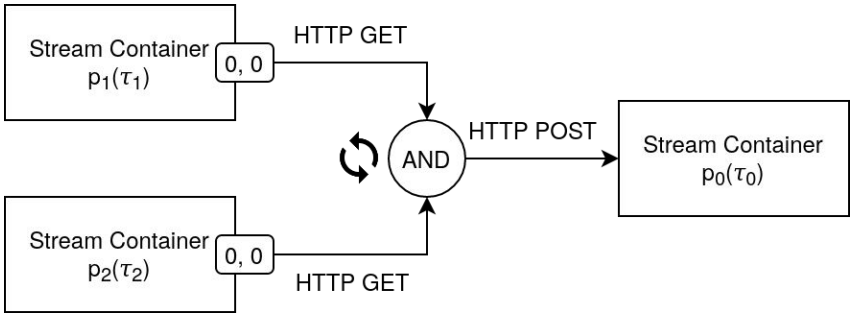
Friedrich-Alexander-Universität
School of Business,
Economics and Society

# Distributed Complex Event Detection with Resource-oriented Stream Containers on the Edge




6th Stream Reasoning Workshop

Daniel Schraudner, Andreas Harth
Chair of Technical Information Systems
Friedrich-Alexander-Universität Erlangen-Nürnberg

This work has been partialy funded by the German Federal Ministry of Education and Research through the MANDAT project (Grant no. 16DTM107A).

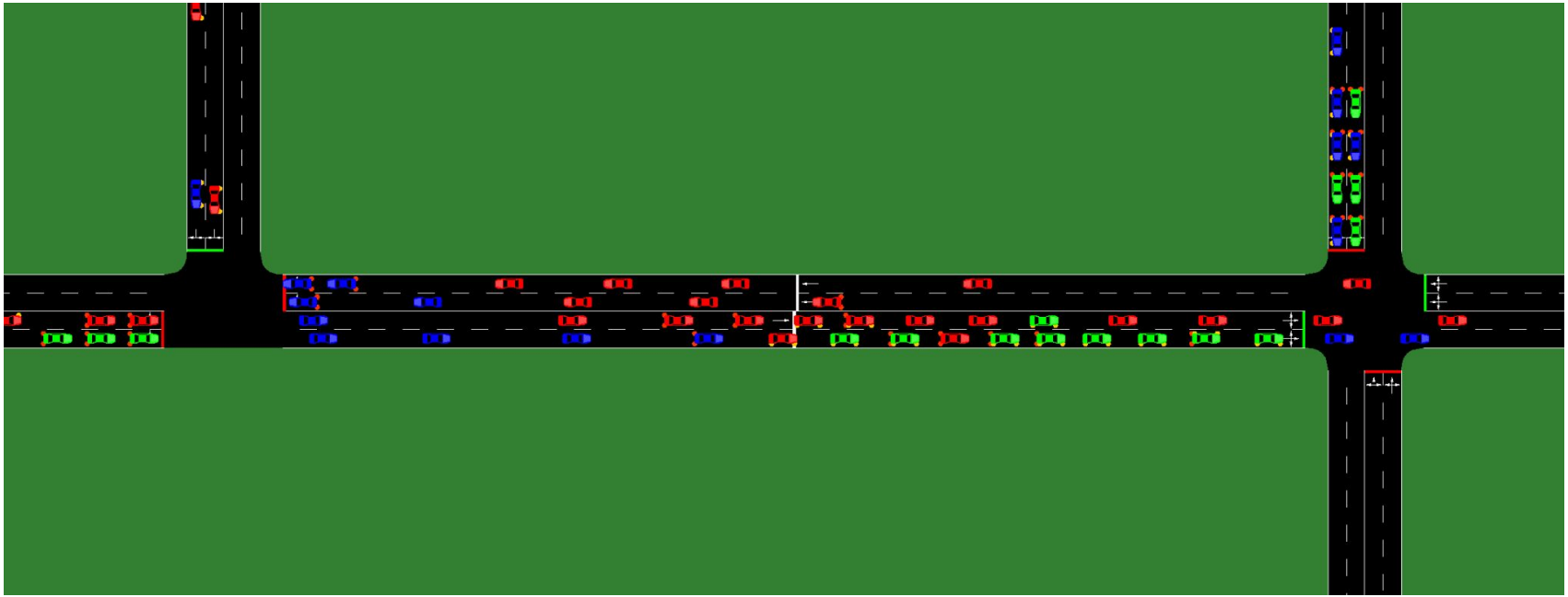
# Agenda

1. Motivation
2. Stream Containers
3. Distributed Complex Event Detection using Stream Containers
4. Splitting up MTL operators
5. Conclusion & Outlook

Friedrich-Alexander-Universität
School of Business,
Economics and Society

# 1. Motivation

# Scenario: Traffic with many Sensors in every Car



Patrik Schneider, Daniel Alvarez-Coello, Anh Le-Tuan, Manh Nguyen Duc, Danh Le Phuoc: Stream Reasoning Playground. ESWC 2022: 406-424

# Sensors & Events

Cars can have lots of sensors, e.g.

- Position
- Speed
- Acceleration
- Cardinal Direction
- Odometer
- Fuel Consumption
- Noise
- Blinker Usage
- Braking
- …

There are lots of interesting events, e.g.

- Red-light Violations
- Turning left/right
- Stopping
- Breakdowns
- Accidents
- Speeding
- U-Turning
- Traffic Congestions
- Motor Problems
- …

# How to detect these Complex Events?

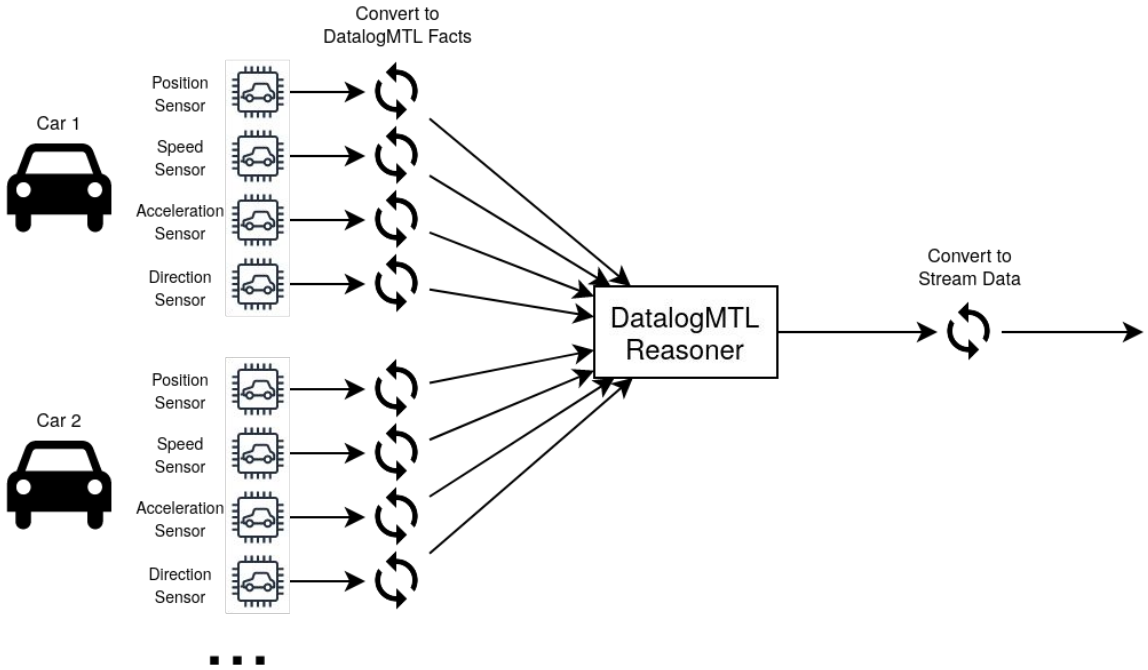
Assumption: We can describe the events using DatalogMTL$^{FP}$ [1]

Example: Breakdown of a car

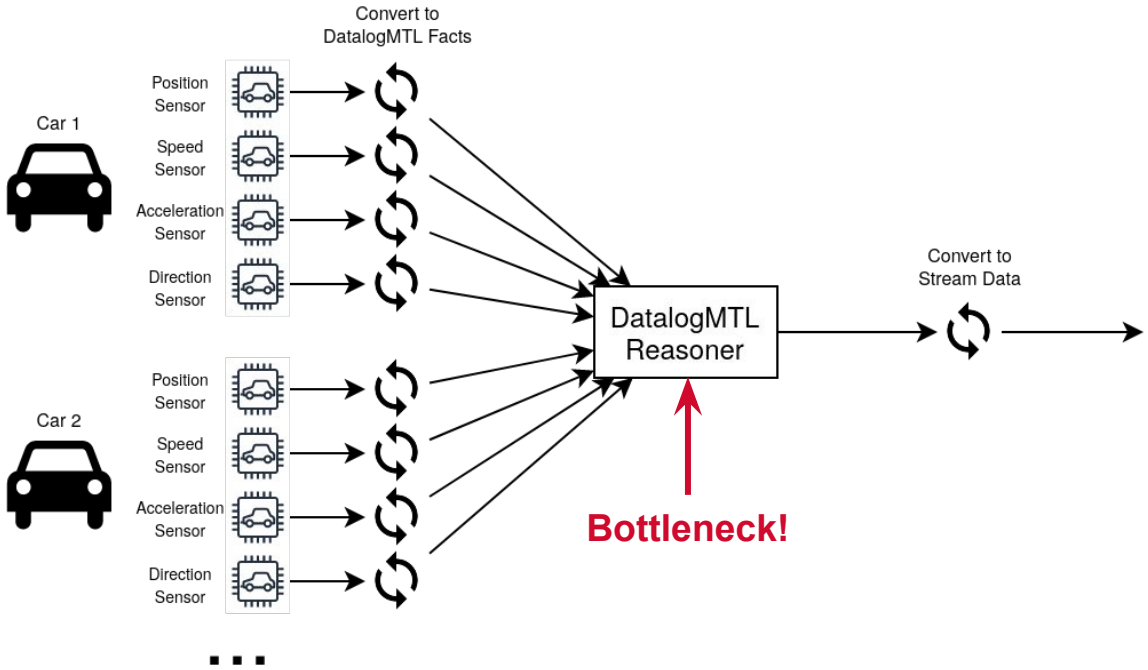
$$breakdown(CAR) \leftarrow \boxminus_{[0,30s]} speed(CAR, 0)$$

[1] Przemyslaw Andrzej Walega, Mark Kaminski, Bernardo Cuenca Grau: Reasoning over Streaming Data in Metric Temporal Datalog. AAAI 2019: 3092-3099

# Streaming Complex Event: Monolithic Approach



# Streaming Complex Event: Monolithic Approach



# 2. Stream Containers

# Stream Containers [1]

- Extension of the Linked Data Platform (LDP; W3C Recommendation) in order to handle stream data in a scalable way and with a clear interface
- LDP provides a RESTful Read-Write Linked Data interface by using containers (similar to directories in a file system)
- New resources can be added to the container using HTTP POST requests

```
@prefix ldp: <http://www.w3.org/ns/ldp#> .

<>      a               ldp:BasicContainer ;
        ldp:contains    </anotherContainer>,
                        </resource> .
```

[1] Daniel Schraudner, Andreas Harth: Stream Containers for Resource-oriented RDF Stream Processing. Stream Reasoning Workshop. 2021

# Containment and Membership Triples

- Containment triple for every contained resource, membership triples depending on conditions

```
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix ex: <http://example.org/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .

<>      a                               ldp:IndirectContainer ;
        ldp:membershipResource          <#car1> ;
        ldp:hasMemberRelation           ex:hasSpeed ;
        ldp:insertedContentRelation     sosa:hasSimpleResult ;
        ldp:contains                    </obs1>,
                                        </obs2> .

<#car1> ex:hasSpeed     75.4,
                        77.7 .
```

```
@prefix ex: <http://example.org/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .

</obs1> a                       sosa:Observation ;
        sosa:observedProperty   ex:speed ;
        sosa:hasSimpleResult    75.4 .
```

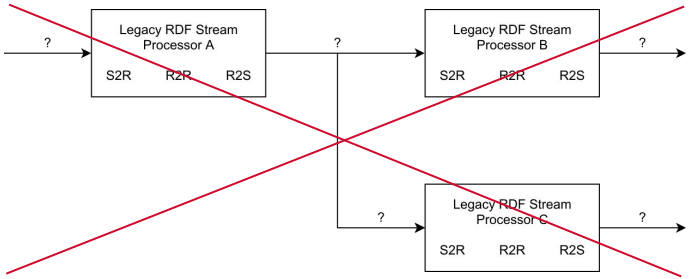
# Stream Containers use Membership Triples for Windows

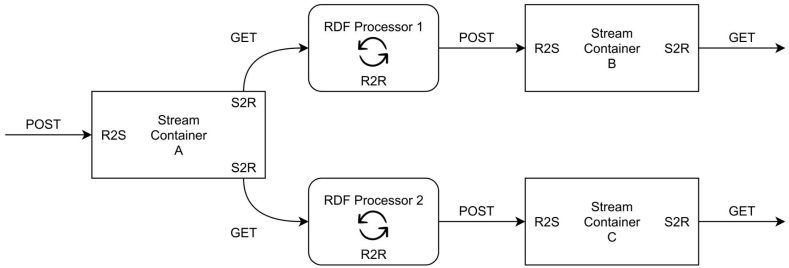
```
@prefix ldpsc: <https://solid.ti.rw.fau.de/public/ns/stream-containers#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://example.org/> .

<>      a               ldpsc:StreamContainer ;
        ldp:contains    </0>,
                        </1>,
                        </2> ;
        ldpsc:window [
                ldp:hasMemberRelation           ex:inWindow ;
                ldp:membershipResource          <#window1> ;
                ldpsc:contentTimestampRelation  sosa:resultTime ;
                ldpsc:beginTime                 "P0S"^^xsd:duration
                ldpsc:endTime                   "PT10S"^^xsd:duration
        ].

<#window1>      ex:inWindow     </1>,
                                </2> .
```

```
@prefix ex: <http://example.org/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .

</2>    a                       sosa:Observation ;
        sosa:observedProperty   ex:speed ;
        sosa:hasSimpleResult    77.7 ;
        sosa:resultTime         "2022-10-17T14:02:08Z"^^xsd:dateTime .
```

# RDF Stream Processing with Stream Containers

- Stream Containers implement the R2S and S2R operators
- Together with agents capable of HTTP requests and RDF Processing (R2R) they can be used for distributed RDF Stream Processing (scalable and with clear interface)
- We were able to show that the semantics of such a system corresponds to the RSP-QL semantics




Daniel Schraudner, Andreas Harth: Stream Containers for Resource-oriented RDF Stream Processing. Stream Reasoning Workshop. 2021

Friedrich-Alexander-Universität
School of Business,
Economics and Society

# 3. Distributed Complex Event Detection using Stream Containers

# Example: Traffic Jam

$$light\_jam(CAR) \leftarrow \diamondsuit_{[0,15]} speed\_less\_than\_equal\_30(CAR)$$

$$medium\_jam(CAR) \leftarrow light\_jam(CAR) \wedge \diamondsuit_{[0,30]} \boxminus_{[0,3]} speed\_less\_than\_equal\_0(CAR)$$

$$heavy\_jam(CAR) \leftarrow light\_jam(CAR) \wedge \boxminus_{[0,30]} \diamondsuit_{[0,10]} \boxminus_{[0,3]} speed\_less\_than\_equal\_0(CAR)$$

$$speed\_less\_than\_equal\_0(CAR) \leftarrow speed(CAR, SPEED) \wedge less\_than\_equal(SPEED, 0)$$

$$speed\_less\_than\_equal\_30(CAR) \leftarrow speed(CAR, SPEED) \wedge less\_than\_equal(SPEED, 30)$$

# Normal Forms

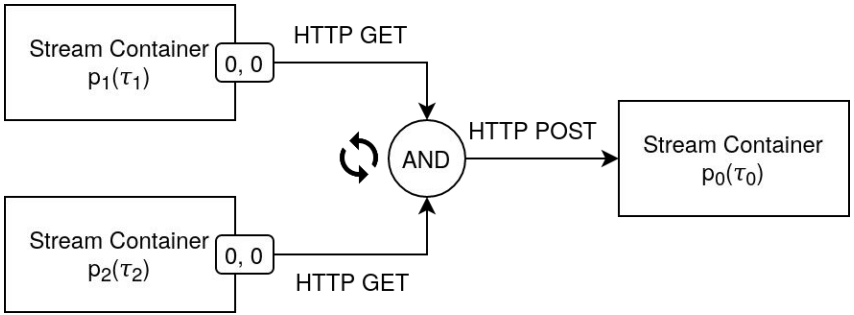
Every program in DatalogMTL$^{FP}$ with can be rewritten to a normal form. The normal form only contains rules of the form:

1. $p_0(\tau_0) \leftarrow p_1(\tau_1) \wedge \cdots \wedge p_n(\tau_n)$
2. $p_0(\tau_0) \leftarrow \boxminus_\sigma p_1(\tau_1)$
3. $p_0(\tau_0) \leftarrow \diamondminus_\sigma p_1(\tau_1)$

# Rules of Type 1

Can be handled by agents that get the current observations (window [0,0]) from Stream Containers and do AND

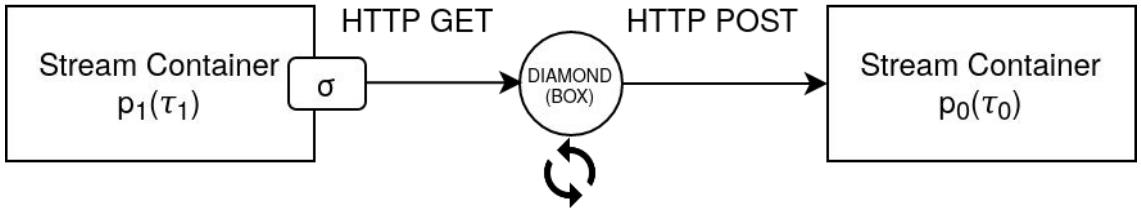
$$p_0(\tau_0) \leftarrow p_1(\tau_1) \wedge \cdots \wedge p_n(\tau_n)$$



# Rules of Types 2 & 3

The handling of Diamond (Box) operator from MTL is shared between Stream Containers and agents: The Stream Container calculates the window, the agent checks if there is a matching observation in the window (all observations in the window are matching).

$$p_0(\tau_0) \leftarrow \Diamond_\sigma p_1(\tau_1) \qquad p_0(\tau_0) \leftarrow \Box_\sigma p_1(\tau_1)$$



# Plan Generation

**Input:**

- DatalogMTL program

**Output:**

- List of Stream Containers with their predicates
- List of agents with their type (And, Box, Diamond), their input Stream Containers and their output Stream Container

# Remarks

- The retention policy of the Stream Containers can be based on the window sizes
- Agents need a global time and a clock (e.g. every second), however, they do not need to be perfectly synchronized as they can use the HTTP Memento protocol to get the window at the correct time
- Use of background knowledge (e.g. `less_than_equal`) and OWL RL reasoning possible

# Remarks

- Individual Stream Containers can be responsible for one predicate or a predicate together with a set of subjects
- It does not matter where individual Stream Containers or agents are situated - could be on the edge, could be in the cloud
- Agents are stateless and thus resilient
- We assume that this systems scales well, especially regarding the program size (different types of events to detect)

Friedrich-Alexander-Universität
School of Business,
Economics and Society

# 4. Splitting up MTL Operators

# Splitting up the MTL Operators

If we generalize the DatalogMTL semantics like this…

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} p(s,o) \text{ iff } p(\nu(s),\nu(o)) \in \mathfrak{M}(t)$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \boxminus_{[w,x]}\psi \text{ iff } \forall min(v,t-w) \geq p \geq max(u,t-x).\mathfrak{M}, p \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \psi$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \diamondminus_{[w,x]}\psi \text{ iff } \exists min(v,t-w) \geq p \geq max(u,t-x).\mathfrak{M} \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \psi$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \boxplus_{[w,x]}\psi \text{ iff } \forall max(u,t+w) \leq p \leq min(v,t+x).\mathfrak{M} \vDash^{\nu}_{[max(u,t+w),min(v,t+x)]} \psi$$

(We get the standard semantics for $u = -\infty, v = +\infty$ )

# Splitting up the MTL Operators

… we can define the LTL and Window operators like this…

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \mathbb{W}^{-}_{[w,x]}\psi \text{ iff } \mathfrak{M}, max(u, t-x) \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \psi$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \mathbb{W}^{+}_{[w,x]}\psi \text{ iff } \mathfrak{M}, max(u, t+w) \vDash^{\nu}_{[max(u,t+w),min(v,t+x)]} \psi$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \Box\psi \text{ iff } \forall t \leq p \leq v.\mathfrak{M}, p \vDash^{\nu}_{[u,v]} \psi$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \Diamond\psi \text{ iff } \exists t \leq p \leq v.\mathfrak{M}, p \vDash^{\nu}_{[u,v]} \psi$$

# Splitting up the MTL Operators

… and show that MTL operators are equivalent to LTL + Window operators.

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \boxminus_{[w,x]} \psi \quad \Leftrightarrow$$

$$\forall min(v, t-w) \geq p \geq max(u, t-x). \mathfrak{M}, p \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \psi \quad \Leftrightarrow$$

$$\forall max(u, t-x) \leq p \leq min(v, t-w). \mathfrak{M}, p \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \psi \quad \Leftrightarrow$$

$$\mathfrak{M}, max(u, t-x) \vDash^{\nu}_{[max(u,t-x),min(v,t-w)]} \Box \psi \quad \Leftrightarrow$$

$$\mathfrak{M}, t \vDash^{\nu}_{[u,v]} \mathbb{W}^{-}_{[w,x]} \Box \psi$$

# 5. Conclusion & Outlook

# Conclusion

We showed how to detect complex events…

- … based on DatalogMTL$^{FP}$ formulas
- … in a distributed and scalable way
- … with a well-established and clear interface (LDP, REST)
- … with a valid approach splitting up MTL operators in LTL and Window operators

# Outlook

- Implementation ongoing
- First results of experiments regarding scalability

# Thank you for your attention!