UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Stream Processing

Lecture 11: Time, Stream Basics, CQL, High-level Streams
31 January, 2018

Foundations of Ontologies and Databases
for Information Systems
CS5130 (Winter 17/18)

Recap: Ontology Change

- Considered ontology change from BR perspective
- Required adaptations and extensions for BR
  - non-classical logics
  - revision of finite belief bases
  - multiple revision
  - iterated revision
- Considered Infinite Iteration and Idea of Formal Learning Theory
- Stabilization/convergence conditions

End of Recap

This Lecture

- Infinite sequences from stream processing perspective
  - Additional aspects: temporality of data, recency, data-driveness, velocity
- Resume OBDA and consider how to lift them to temporal OBDA and streaming OBDA
  - Temporal OBDA: Add time aspect (somewhere)
  - Stream OBDA: Higher-level stream w.r.t. ontology (and mappings)

Temporalized OBDA

A Confession

- Ontology-Based Data Access on temporal and **Streaming** Data
- But: Streams are temporal streams and we talk about local temporal reasoning

Adding a Temporal Dimension to OBDA

- Most conservative strategy: handle time as "ordinary" attribute *time*

$$\left\{ \begin{array}{c} meas(x) \wedge \\ val(x,y) \wedge \\ time(x,z) \end{array} \right\} \longleftarrow$$

```
SELECT f(MID) AS m, Mval AS y, MtimeStamp AS z
FROM MEASUREMENT
```

- Classical Mapping
- Pro: Minimal (no) adaptation
- Contra:
  - No control on "logic of time"
  - Need reification
    - sometimes necessary (because DLs provided only predicates up to arity 2)
    - but not that "natural"

Flow of Time

- Flow of time $(T, \leq_T)$ is a structure with a time domain T and a binary relation $\leq_T$ over it.
- Flow metaphor hints on directionality and dynamic aspect of time
- But still different forms of flow are possible

- One can consider concrete structures of flow of (time), as done here
- Or investigate them additionally axiomatically
- An early model-theoretic and axiomatic treatise:

  Lit: J. van Benthem. The Logic of Time: A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse. Reidel, 2. edition, 1991.

The Family of Flows of Time

- Domain T
  - points (atomic time instances)
  - pairs of points (application time, transaction time)
  - intervals etc.
- Properties of the time relation $\leq_T$
  - Non-branching (linear) vs. branching
    Linearity:
    - reflexive: $\forall t \in T$: $t \leq_T t$
    - antisymmetric: $\forall t_1, t_2 \in T$: $(t_1 \leq t_2 \wedge t_2 \leq_T t_1) \Rightarrow t_1 = t_2$
    - transitive: $\forall t_1, t_2, t_3 \in T : (t_1 \leq_T t_2 \wedge t_2 \leq t_3) \Rightarrow t_1 \leq t_3$.
    - total: $\forall t_1, t_2 \in T$: $t_1 \leq t_2 \vee t_2 \leq t_1 \vee t_1 = t_2$.
- Existence of first or last element
- discreteness (Example: $T = \mathbb{N}$); also used for modeling state sequences;
- density (Example: $T = \mathbb{Q}$);
- continuity (Example: $T = \mathbb{R}$)

The Family of Flows of Time

- Domain T
  - points (atomic time instances)
  - pairs of points (application time, transaction time)
  - intervals etc.
- Properties of the time relation $\leq_T$
  - Non-branching (linear) vs. branching
    Linearity:
    - reflexive: $\forall t \in T$: $t \leq_T t$
    - antisymmetric: $\forall t_1, t_2 \in T$: $(t_1 \leq t_2 \wedge t_2 \leq_T t_1) \Rightarrow t_1 = t_2$
    - transitive: $\forall t_1, t_2, t_3 \in T : (t_1 \leq_T t_2 \wedge t_2 \leq t_3) \Rightarrow t_1 \leq t_3$.
    - total: $\forall t_1, t_2 \in T$: $t_1 \leq t_2 \vee t_2 \leq t_1 \vee t_1 = t_2$.
- Existence of first or last element
- discreteness (Example: $T = \mathbb{N}$); also used for modeling state sequences;
- density (Example: $T = \mathbb{Q}$);
- continuity (Example: $T = \mathbb{R}$)

The Family of Flows of Time

- Domain T
  - points (atomic time instances)
  - pairs of points (application time, transaction time)
  - intervals etc.
- Properties of the time relation $\leq_T$
  - Non-branching (linear) vs. branching
    Linearity:
    - reflexive: $\forall t \in T$: $t \leq_T t$
    - antisymmetric: $\forall t_1, t_2 \in T$: $(t_1 \leq t_2 \wedge t_2 \leq_T t_1) \Rightarrow t_1 = t_2$
    - transitive: $\forall t_1, t_2, t_3 \in T : (t_1 \leq_T t_2 \wedge t_2 \leq t_3) \Rightarrow t_1 \leq t_3$.
    - total: $\forall t_1, t_2 \in T$: $t_1 \leq t_2 \vee t_2 \leq t_1 \vee t_1 = t_2$.
- Existence of first or last element
- discreteness (Example: $T = \mathbb{N}$); also used for modeling state sequences;
- density (Example: $T = \mathbb{Q}$);
- continuity (Example: $T = \mathbb{R}$)

The Family of Flows of Time

- Domain T
  - points (atomic time instances)
  - pairs of points (application time, transaction time)
  - intervals etc.
- Properties of the time relation $\leq_T$
  - Non-branching (linear) vs. branching
    Linearity:
    - reflexive: $\forall t \in T$: $t \leq_T t$
    - antisymmetric: $\forall t_1, t_2 \in T$: $(t_1 \leq t_2 \wedge t_2 \leq_T t_1) \Rightarrow t_1 = t_2$
    - transitive: $\forall t_1, t_2, t_3 \in T : (t_1 \leq_T t_2 \wedge t_2 \leq t_3) \Rightarrow t_1 \leq t_3$.
    - total: $\forall t_1, t_2 \in T$: $t_1 \leq t_2 \vee t_2 \leq t_1 \vee t_1 = t_2$.
- Existence of first or last element
- discreteness (Example: $T = \mathbb{N}$); also used for modeling state sequences;
- density (Example: $T = \mathbb{Q}$);
- continuity (Example: $T = \mathbb{R}$)

Temporalized OBDA: General Approach

- Semantics rests on family of interpretations $(\mathcal{I}_t)_{t \in T}$
- Temporal ABox $\tilde{A}$: Finite set of T-tagged ABox axioms

Example

$val(s_0, 90^\circ)\langle 3s \rangle$ holds in $(\mathcal{I}_t)_{t \in T}$ iff $\mathcal{I}_{3s} \models val(s_0, 90^\circ)$
"sensor s_0 has value 90° at time point 3s"

- Alternative sequence representation of temporal ABox $\tilde{\mathcal{A}}$
  - $(\mathcal{A}_t)_{t \in T'}$ (where T' are set of timestamps in T)
  - $\mathcal{A}_t = \{ax \mid ax\langle t \rangle \in \tilde{\mathcal{A}}\}$

Definition (Adapted notion of OBDA rewriting)

$$cert(Q, (Sig, \mathcal{T}, (\mathcal{A}_t)_{t \in T'}) = ans(Q_{rew}, (DB(\mathcal{A}_t))_{t \in T'})$$

Temporalized OBDA:TCQs

- Different approaches based on modal (temporal) operators
- LTL operators only in QL (Borgwardt et al. 13)

Example

$$\begin{aligned} \mathit{Critical}(x) &= \exists y.\mathit{Turbine}(x) \wedge \mathit{showsMessage}(x,y) \wedge \\ & \quad \mathit{FailureMessage}(y) \\ Q(x) &= \bigcirc^{-1}\bigcirc^{-1}\bigcirc^{-1}(\Diamond(\mathit{Critical}(x) \wedge \bigcirc\Diamond\mathit{Critical}(x))) \end{aligned}$$

"turbine has been at least two times in a critical situation in the last three time units"

- CQ embedded into LTL template
- Special operators taking care of endpoints of state sequencing
- Not well-suited for OBDA as non-safe
- Rewriting simple due to atemporal TBox

Lit: S. Borgwardt, M. Lippmann, and V. Thost. Temporal query answering in the description logic dl-lite. In FroCs, volume 8152 of LNCS, pages 165–180, 2013.

Temporalized OBDA: TQL

- LTL operators in TBox and T argument in QL

Example

$$\begin{array}{rcl} \text{TBox axiom} & : & \mathit{showsAnomaly} \sqsubseteq \Diamond \mathit{UnplanedShutDown} \\ & & \text{``if turbine shows anomaly (now)} \\ & & \text{then sometime in the future it will shut down''} \\ \text{Query} & : & \exists t.3s \leq t \leq 6s \land \mathit{showsAnomaly}(x, t) \end{array}$$

- Can formulate rigidity assumptions
- Rewriting not trivial

Lit: A. Artale, R. Kontchakov, F. Wolter, and M. Zakharyaschev. Temporal description logic for ontology- based data access. In IJCAI'13, pages 711–717. AAAI Press, 2013.

Stream Basics

Streams

Definition (Stream)

A stream S is a potentially infinite sequence of objects d over some domain D.

- "Streams are forever"

  Lit: J. Endrullis, D. Hendriks, and J. W. Klop. Streams are forever. Bulletin of the EATCS, 109:70–106, 2013.

- "Order matters!"

  Lit: E. D. Valle et al. Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web, 4(2):219–231, 2013.

- "It's a streaming world!"

  Lit: E. Della Valle, et al. It?s a streaming world! Reasoning upon rapidly changing information. Intelligent Systems, IEEE, 24(6):83–89, nov.-dec. 2009.

Streams

Definition (Stream)

A stream S is a potentially infinite sequence of objects d over some domain D.

- "Streams are forever"

  Lit: J. Endrullis, D. Hendriks, and J. W. Klop. Streams are forever. Bulletin of the EATCS, 109:70–106, 2013.

- "Order matters!"

  Lit: E. D. Valle et al. Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web, 4(2):219–231, 2013.

- "It's a streaming world!"

  Lit: E. Della Valle, et al. It?s a streaming world! Reasoning upon rapidly changing information. Intelligent Systems, IEEE, 24(6):83–89, nov.-dec. 2009.

Streams

Definition (Stream)

A stream S is a potentially infinite sequence of objects d over some domain D.

- "Streams are forever"

  Lit: J. Endrullis, D. Hendriks, and J. W. Klop. Streams are forever. Bulletin of the EATCS, 109:70–106, 2013.

- "Order matters!"

  Lit: E. D. Valle et al. Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web, 4(2):219–231, 2013.

- "It's a streaming world!"

  Lit: E. Della Valle, et al. It?s a streaming world! Reasoning upon rapidly changing information. Intelligent Systems, IEEE, 24(6):83–89, nov.-dec. 2009.

Streams

Definition (Stream)

A stream S is a potentially infinite sequence of objects d over some domain D.

- "Streams are forever"

  Lit: J. Endrullis, D. Hendriks, and J. W. Klop. Streams are forever. Bulletin of the EATCS, 109:70–106, 2013.

- "Order matters!"

  Lit: E. D. Valle et al. Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web, 4(2):219–231, 2013.

- "It's a streaming world!"

  Lit: E. Della Valle, et al. It?s a streaming world! Reasoning upon rapidly changing information. Intelligent Systems, IEEE, 24(6):83–89, nov.-dec. 2009.

Adding a Time Dimension

Definition (Temporal Stream)

A temporal stream S is a potentially infinite sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

- Consider non-branching (or: linear) time, i.e., $\leq_T$ is
- We assume that there is no last element in T
- We do not restrict T further, so it may be
  - discrete or
  - dense or
  - continuous

Adding a Time Dimension

Definition (Temporal Stream)

A temporal stream S is a potentially infinite sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

- Consider non-branching (or: linear) time, i.e., $\leq_T$ is
- We assume that there is no last element in T
- We do not restrict T further, so it may be
  - discrete or
  - dense or
  - continuous

Arrival Ordering

- Sequence fixed by arrival ordering fixed $<_{ar}$
- $<_{ar}$ is a strict total ordering on the elements of S
- Synchronous streams: $\leq_T$ compatible with $<_{ar}$
- Compatibility: For all $d_1\langle t_1\rangle, d_2\langle t_2\rangle \in S$: If $d_1\langle t_1\rangle <_{ar} d_2\langle t_2\rangle$, then $t_1 \leq_T t_2$.
- Asynchronous streams: $\leq_T$ not necessarily compatible with $<_{ar}$

Convention for the following

- Consider only temporal streams
- Consider only synchronous streams $\Longrightarrow$ neglect $<_{ar}$.
- Represent streams as a potentially infinite multi-set (bag) of elements

Arrival Ordering

- Sequence fixed by arrival ordering fixed $<_{ar}$
- $<_{ar}$ is a strict total ordering on the elements of S
- Synchronuous streams: $\leq_T$ compatible with $<_{ar}$
- Compatibility: For all $d_1\langle t_1\rangle, d_2\langle t_2\rangle \in S$: If $d_1\langle t_1\rangle <_{ar} d_2\langle t_2\rangle$, then $t_1 \leq_T t_2$.
- Asynchronous streams: $\leq_T$ not necessarily compatible with $<_{ar}$

Convention for the following

- Consider only temporal streams
- Consider only synchronous streams $\Longrightarrow$ neglect $<_{ar}$.
- Represent streams as a potentially infinite multi-set (bag) of elements

Arrival Ordering

- Sequence fixed by arrival ordering fixed $<_{ar}$
- $<_{ar}$ is a strict total ordering on the elements of S
- Synchronuous streams: $\leq_T$ compatible with $<_{ar}$
- Compatibility: For all $d_1\langle t_1\rangle, d_2\langle t_2\rangle \in S$: If $d_1\langle t_1\rangle <_{ar} d_2\langle t_2\rangle$, then $t_1 \leq_T t_2$.
- Asynchronous streams: $\leq_T$ not necessarily compatible with $<_{ar}$

Convention for the following

- Consider only temporal streams
- Consider only synchronous streams $\Longrightarrow$ neglect $<_{ar}$.
- Represent streams as a potentially infinite multi-set (bag) of elements

Stream Stack and Stream Research

- Low-level sensor perspective (semantic sensor networks)
  - Develop fast algorithms on high-frequency streams with minimal space consumption
  - Considers approximate algorithms for aggregation functions
  - See lecture "Non-standard DBs" by Ralf Möller
- Data stream management system (DSMS) perspective
  - Provide whole DB systems for streams of structured (relational) data
  - Deals with all aspects relevant in static DBMS adapted to stream scenario
  - See lecture "Non-standard DBs" by Ralf Möller and this lecture
  - Stream Query Language
- High-level and Ontology layer streams
  - Processing stream of assertions (RDF triples) w.r.t. an ontology
  - Related: Complex Event Processing (CEP)
  - this and next lecture

Stream Stack and Stream Research

- Low-level sensor perspective (semantic sensor networks)
  - Develop fast algorithms on high-frequency streams with minimal space consumption
  - Considers approximate algorithms for aggregation functions
  - See lecture "Non-standard DBs" by Ralf Möller

- Data stream management system (DSMS) perspective
  - Provide whole DB systems for streams of structured (relational) data
  - Deals with all aspects relevant in static DBMS adapted to stream scenario
  - See lecture "Non-standard DBs" by Ralf Möller and this lecture
  - Stream Query Language

- High-level and Ontology layer streams
  - Processing stream of assertions (RDF triples) w.r.t. an ontology
  - Related: Complex Event Processing (CEP)
  - this and next lecture

Stream Stack and Stream Research

- Low-level sensor perspective (semantic sensor networks)
  - Develop fast algorithms on high-frequency streams with minimal space consumption
  - Considers approximate algorithms for aggregation functions
  - See lecture "Non-standard DBs" by Ralf Möller

- Data stream management system (DSMS) perspective
  - Provide whole DB systems for streams of structured (relational) data
  - Deals with all aspects relevant in static DBMS adapted to stream scenario
  - See lecture "Non-standard DBs" by Ralf Möller and this lecture
  - Stream Query Language

- High-level and Ontology layer streams
  - Processing stream of assertions (RDF triples) w.r.t. an ontology
  - Related: Complex Event Processing (CEP)
  - this and next lecture

Local vs. Global Stream Processing

- Global aim: Learn about the whole by looking at the parts
  - Examples: inductive learning, ontology change, iterated belief revision (see slides before), robotics oriented stream processing with plan generation
  - May produce also an output stream
  - But in the end the whole stream counts

- Local aim: Monitor window contents with time-local
  - Examples: Real-time monitoring, simulation for reactive diagnostics

- Categories not exclusive
  - In learning one applies operation on (NOW)-window to learn about stream
  - In predictive analytics one monitors with window in order to predict upcoming events

Domain Objects for Streams

Definition (Temporal Stream)

A stream S is a sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

Streamified OBDA has to deal with different types of domains

Domain Objects for Streams

Definition (Temporal Stream)

A stream S is a sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

Streamified OBDA has to deal with different types of domains

D_1 = a set of typed relational tuples adhering to a relational schema

- Streams at the backend sources
- $S_{rel} = \{(s_1, 90^\circ)\langle 1s\rangle, (s_2, 92^\circ)\langle 2s\rangle, (s_1, 94^\circ)\langle 3s\rangle, \dots\}$
- Schema: hasSensorRelation(Sensor:string, temperature:integer)

Domain Objects for Streams

Definition (Temporal Stream)

A stream S is a sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

Streamified OBDA has to deal with different types of domains

D_2 = set of untyped tuples (of the same arity)

- Stream of tuples resulting as bindings for subqueries

Domain Objects for Streams

Definition (Temporal Stream)

A stream S is a sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

Streamified OBDA has to deal with different types of domains

D_3 = set of assertions (RDF tuples).

- $S_{rdf} = \{\ val(s_0, 90^\circ)\langle 1s\rangle, val(s_2, 92^\circ)\langle 2s\rangle,\ val(s_1, 94^\circ)\langle 3s\rangle, \ldots\ \}$

Domain Objects for Streams

Definition (Temporal Stream)

A stream S is a sequence of timestamped objects $d\langle t\rangle$ over some domain D and flow of time $(T, \leq_T)$.

Streamified OBDA has to deal with different types of domains

D_4 = set of RDF graphs

Taming the Infinite

Nearly all stream processing approaches provide a fundamental means to cope with potential infinity of streams, namely ...

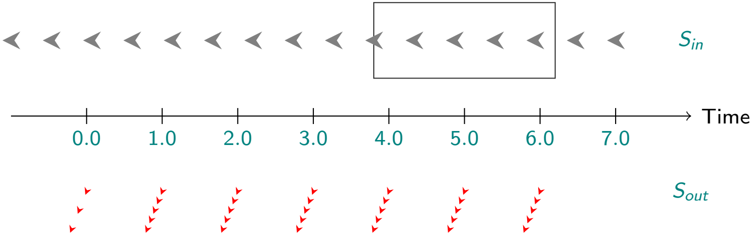
Taming the Infinite

Nearly all stream processing approaches provide a fundamental means to cope with potential infinity of streams, namely ...

- Stream query continuous, not one-shot activity
- Window content continuosly updated

Taming the Infinite

Nearly all stream processing approaches provide a fundamental means to cope with potential infinity of streams, namely ...



- Here a time-based window of width 3 seconds
- and slide 1 second is applied

Window Operators: Classification

- Direction of movement of the endpoints
  - Both endpoints fixed (needed for "historical" queries)
  - Both moving/sliding
  - One moving the other not
- Window size
  - Temporal
  - Tuple-based
  - Partitioned window
  - Predicate window
- Window update
  - tumbling
  - sampling
  - overlapping

Window Operators: Classification

- Direction of movement of the endpoints
  - Both endpoints fixed (needed for "historical" queries)
  - Both moving/sliding
  - One moving the other not
- Window size
  - Temporal
  - Tuple-based
  - Partitioned window
  - Predicate window
- Window update
  - tumbling
  - sampling
  - overlapping

Window Operators: Classification

- Direction of movement of the endpoints
  - Both endpoints fixed (needed for "historical" queries)
  - Both moving/sliding
  - One moving the other not

- Window size
  - Temporal
  - Tuple-based
  - Partitioned window
  - Predicate window

- Window update
  - tumbling
  - sampling
  - overlapping

Why is the Window Concept so Important?

- We give an answer using the word perspective on stream processing according to (Gurevich et al. 07)
- Streams = finite or infinite words over an alphabet (domain) D
  - D^* = finite words over D
  - $D^\omega =$ infinite (ω-) words over D
  - $D^\infty =$ finite and infinite words over D
  - $\circ =$ word concatenation (usually not mentioned)
- Stream operators Q are functions/queries of the form

$$Q : D_1^\infty \longrightarrow D_2^\infty$$

- Assume w.l.o.g that $D_1 = D_2 = D$.

Lit: Y. Gurevich, D. Leinders, and J. Van Den Bussche. A theory of stream queries. In Proceedings of the 11th International Conference on Database Programming Languages, DBPL'07, pages 153–168, 2007.

Genuine Streams are Finite Prefix Determined

- Open ball around u:
  $B(u) := uD^\infty = \{s \in D^\infty \mid \text{There is } s' \in D^\infty \text{ s.t. } s = u \circ s'\}$

Definition (Axiom of finite prefix determinedness (FP))

For all $s \in D^\infty$ and all $u \in D^*$:
If $Q(s) \in uD^\infty$, there is $w \in D^*$ s.t. $s \in wD^\infty \subseteq Q^{-1}(uD^\infty)$

- (FP) expresses a continuity condition w.r.t. a topology

Genuine Streams are Finite Prefix Determined

- Open ball around u:
  $B(u) := uD^\infty = \{s \in D^\infty \mid \text{There is } s' \in D^\infty \text{ s.t. } s = u \circ s'\}$

Definition (Axiom of finite prefix determinedness (FP))

For all $s \in D^\infty$ and all $u \in D^*$:
If $Q(s) \in uD^\infty$, there is $w \in D^*$ s.t. $s \in wD^\infty \subseteq Q^{-1}(uD^\infty)$

- (FP) expresses a continuity condition w.r.t. a topology
- Reminder: A topology is a structure $(X, \mathcal{O})$ where
  - $\mathcal{O} \subseteq Pow(X)$
  - $\emptyset, X \in \mathcal{O}$
  - $\mathcal{O}$ closed under finite intersections
  - $\mathcal{O}$ closed under arbitrary unions
- A basis for $\mathcal{O}$ is a set $\mathcal{B} \subseteq Pow(X)$ s.t.: Every $S \in \mathcal{O}$ is a union of elements of $\mathcal{B}$.

Genuine Streams are Finite Prefix Determined

- Open ball around u:
  $B(u) := uD^\infty = \{s \in D^\infty \mid \text{There is } s' \in D^\infty \text{ s.t. } s = u \circ s'\}$

Definition (Axiom of finite prefix determinedness (FP))

For all $s \in D^\infty$ and all $u \in D^*$:
If $Q(s) \in uD^\infty$, there is $w \in D^*$ s.t. $s \in wD^\infty \subseteq Q^{-1}(uD^\infty)$

- (FP) expresses a continuity condition w.r.t. a topology
- Gurevich topology
  $\mathcal{T}_G = (D^\infty, \{AD^\infty \mid A \subseteq D^*\})$
- $B(u)$ for $u \in D^*$ are basis for $\mathcal{T}_G$.
- A function $Q : D^\infty \longrightarrow D^\infty$ is continuous iff for every open ball B: $Q^{-1}(B)$ is open.
- i.e., iff Q fulfills (FP)

Abstract Computability

- For $K : D^* \longrightarrow D^*$ (window function)
- Repeated application of K

$$\begin{aligned} Repeat(K) : \quad D^\infty &\longrightarrow D^\infty \\ s &\mapsto \bigcirc_{i=0}^{length(s)} K(s^{\leq i}) \end{aligned}$$

Definition (Gurevich et al. 2007)

K is a kernel for Q iff $Q = Repeat(K)$.

Q is abstract computable (AC) iff it has a kernel.

Abstract Computability

- For $K : D^* \longrightarrow D^*$ (window function)
- Repeated application of K

$$\begin{aligned} Repeat(K): \quad D^\infty &\longrightarrow D^\infty \\ s &\mapsto \bigcirc_{i=0}^{length(s)} K(s^{\leq i}) \end{aligned}$$

Definition (Gurevich et al. 2007)

K is a kernel for Q iff $Q = Repeat(K)$.

Q is abstract computable (AC) iff it has a kernel.

Abstract Computability

- For $K : D^* \longrightarrow D^*$ (window function)
- Repeated application of K

$$\begin{array}{rccc} Repeat(K): & D^\infty & \longrightarrow & D^\infty \\ & s & \mapsto & \bigcirc_{i=0}^{length(s)} K(s^{\leq i}) \end{array}$$

Definition (Gurevich et al. 2007)

K is a kernel for Q iff $Q = Repeat(K)$.

Q is abstract computable (AC) iff it has a kernel.

A Representation Theorem

Theorem

The set of AC functions are exactly those stream functions fulfilling FP (i.e. that are continuous) and mapping finite streams to finite streams

- Further interesting representation results by considering restrictions on window
- Gurevich et al. also describe computation model (abstract state machines)

A Representation Theorem

Theorem

The set of AC functions are exactly those stream functions fulfilling FP (i.e. that are continuous) and mapping finite streams to finite streams

- Further interesting representation results by considering restrictions on window
- Gurevich et al. also describe computation model (abstract state machines)

Example for non-continuos stream functions

Example

Query CHECK

- $a, b \in D$
- CHECK(s) = (a) if b does not occur in s
- Otherwise $CHECK(s) = ()$ = empty stream

- CHECK is not continuous (and hence not an AC function):
  - Consider open ball $B(a)$.
  - $() \in CHECK^{-1}(B(a))$
  - But the only open ball containing $()$ is $B(()) = D^\infty$
  - But $B(()) \nsubseteq CHECK^{-1}(B(a))$ because
  - $CHECK(b) = () \notin B(a)$

Relational Stream Processing with CQL

Relational Data Stream Processing

- Different groups working on DSMS around 2004
  - Academic prototypes: STREAM and CQL (Stanford); TelgraphCQ (Berkeley) (extends PostGreSQL); Aurora/Borealis (Brandeis, Brown and MIT); PIPES from Marburg University
  - Commercial systems: StreamBase, Truviso (Standalone), extensions of commercial DBMS (MySQL, PostgreSQL, DB2 etc.)
- Though well investigated and many similarities there is no streaming SQL standard
- First try for standardization:

  Lit: N. Jain et al. Towards a streaming sql standard. Proc. VLDB Endow., 1(2):1379–1390, Aug. 2008.
- But if development speed similar to that for introducing temporal dimension into SQL, then we have to wait...

Relational Data Stream Processing

- Different groups working on DSMS around 2004
  - Academic prototypes: STREAM and CQL (Stanford); TelgraphCQ (Berkeley) (extends PostGreSQL); Aurora/Borealis (Brandeis, Brown and MIT); PIPES from Marburg University
  - Commercial systems: StreamBase, Truviso (Standalone), extensions of commercial DBMS (MySQL, PostgreSQL, DB2 etc.)

- Though well investigated and many similarities there is no streaming SQL standard
- First try for standardization:

  Lit: N. Jain et al. Towards a streaming sql standard. Proc. VLDB Endow., 1(2):1379–1390, Aug. 2008.
- But if development speed similar to that for introducing temporal dimension into SQL, then we have to wait...

CQL (Continuous Query Language)

- Early relational stream query language extending SQL
- Developed in Stanford as part of a DSMS called STREAM
- Semantics theoretically specified by denotational semantics
- Practically, development of CQL was accompanied by the development the Linear Road Benchmark (LRB) (http://www.cs.brandeis.edu/~linearroad/)
- Had immense impact also on development of early RDF streaming engines in RSP community https://www.w3.org/community/rsp/)

Lit: A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. The VLDB Journal, 15:121–142, 2006.

Lit: A. Arasu et al. Linear road: A stream data management benchmark. In VLDB, pages 480–491, 2004.

CQL Operators

- Special data structure next to streams: relations R
  - R maps times t to ordinary (instantaneous) relations $R(t)$
  - Motivation: Use of ordinary SQL operators on instantaneous relations
- Operators
  - Stream-to-relation = window operator
  - Relation-to-relation = standard SQL operators at every single time point
  - relation-stream = for getting streams agains
- Non-predictability condition for operators op:
  - If two inputs S_1 , S_2 are the same up to t, then $op(S_1)(t) = op(S_2)(t)$.
    (This is related to (FP))

CQL Windows

- Window operators are stream-to-relation operators
- CQL knows tuple-based, partition based, and time-based windows

Definition (Semantics of Window Operator)

R = S [Range wr Slide sl]

- with slide parameter sl and range wr
- $t_{start} = \lfloor t/sl \rfloor \cdot sl$
- $t_{end} = max\{t_{start} - wr, 0\}$

$$R(t) = \begin{cases} \emptyset & \text{if } t < sl \\ \{s \mid s\langle t' \rangle \in S \text{ and } t_{end} \leq t' \leq t_{start}\} & \text{else} \end{cases}$$

- Standard slide = 1: [RANGE wr]
- Left end fixed: [Range UNBOUND]
- Width 0: [NOW]

CQL Windows

- Window operators are stream-to-relation operators
- CQL knows tuple-based, partition based, and time-based windows

Definition (Semantics of Window Operator)

R = S [Range wr Slide sl]

- with slide parameter sl and range wr
- $t_{start} = \lfloor t/sl \rfloor \cdot sl$
- $t_{end} = max\{t_{start} - wr, 0\}$

$$R(t) = \begin{cases} \emptyset & \text{if } t < sl \\ \{s \mid s\langle t' \rangle \in S \text{ and } t_{end} \leq t' \leq t_{start}\} & \text{else} \end{cases}$$

- Standard slide = 1: [RANGE wr]
- Left end fixed: [Range UNBOUND]
- Width 0: [NOW]

CQL Windows

- Window operators are stream-to-relation operators
- CQL knows tuple-based, partition based, and time-based windows

Definition (Semantics of Window Operator)

R = S [Range wr Slide sl]

- with slide parameter sl and range wr
- $t_{start} = \lfloor t/sl \rfloor \cdot sl$
- $t_{end} = max\{t_{start} - wr, 0\}$

$$R(t) = \begin{cases} \emptyset & \text{if } t < sl \\ \{s \mid s\langle t' \rangle \in S \text{ and } t_{end} \leq t' \leq t_{start}\} & \text{else} \end{cases}$$

- Standard slide = 1: [RANGE wr]
- Left end fixed: [Range UNBOUND]
- Width 0: [NOW]

Sliding Window Example in CQL

- Flow of time $(\mathbb{N}, \leq)$
- Input stream

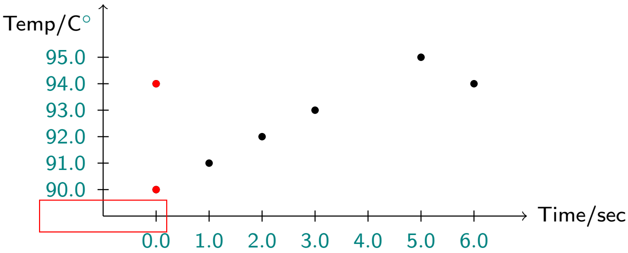
$$S = \{(s_0, 90^\circ)\langle 0\rangle, (s_1, 94^\circ)\langle 0\rangle, (s_0, 91^\circ)\langle 1\rangle, (s_0, 92^\circ)\langle 2\rangle, (s_0, 93^\circ)\langle 3\rangle, (s_0, 95^\circ)\langle 5\rangle, (s_0, 94^\circ)\langle 6\rangle \ldots\}$$

- Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93), (s_0, 95)\}$	$\{(s_0, 93), (s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

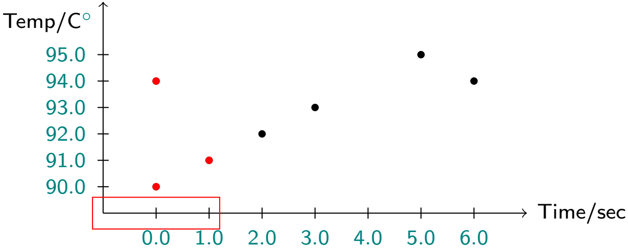


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

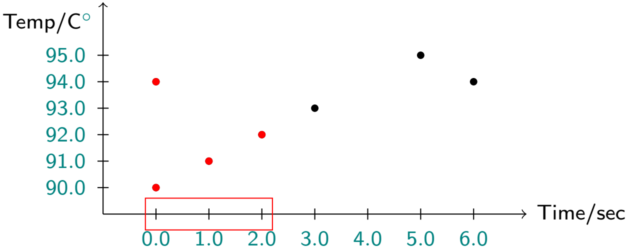


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

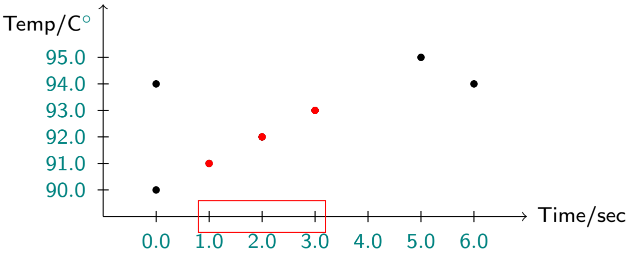


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

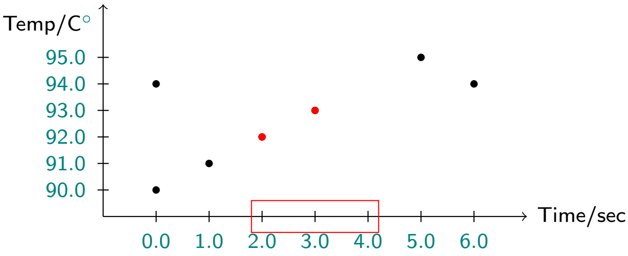


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

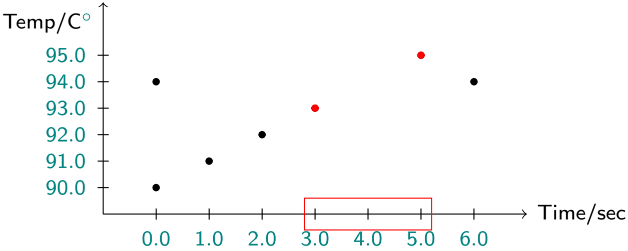


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$

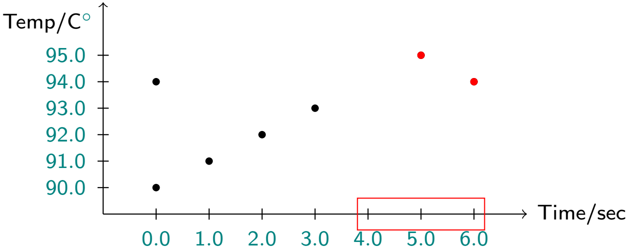


Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Sliding Window Example in CQL

$S = \{(s_0, 90)\langle 0\rangle, (s_1, 94)\langle 0\rangle, (s_0, 91)\langle 1\rangle, (s_0, 92)\langle 2\rangle, (s_0, 93)\langle 3\rangle, (s_0, 95)\langle 5\rangle, (s_0, 94)\langle 6\rangle\}$



Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 93), (s_0, 95)\}$	$\{(s_0, 95), (s_0, 94)\}$

Relation vs. Stream

$$S = \{(s_0, 90^\circ)\langle 0\rangle, (s_1, 94^\circ)\langle 0\rangle, (s_0, 91^\circ)\langle 1\rangle, (s_0, 92^\circ)\langle 2\rangle, (s_0, 93^\circ)\langle 3\rangle, (s_0, 95^\circ)\langle 5\rangle, (s_0, 94^\circ)\langle 6\rangle \ldots.\}$$

- Output relation $R =$ `S [Range 2 Slide 1]`

t :	0	1	2	3	4	5	6
$R(t)$:	$\{(s_0, 90), (s_1, 94)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91)\}$	$\{(s_0, 90), (s_1, 94), (s_0, 91), (s_0, 92)\}$	$\{(s_0, 91), (s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93)\}$	$\{(s_0, 92), (s_0, 93), (s_0, 95)\}$	$\{(s_0, 93), (s_0, 95), (s_0, 94)\}$

- Note that there are also entries for second 4
- Note that timestamps are lost in the bags
- Slides are local to streams and may be different over different streams

Relation-To-Stream Operators

- Output stream of input relation R:

$$
\begin{aligned}
\mathit{Istream}(R) &= \bigcup_{t \in T} (R(t) \setminus R(t-1)) \times \{t\} \\
&\quad \text{stream of inserted elements} \\
\mathit{Dstream}(R) &= \bigcup_{t \in T} (R(t-1) \setminus R(t)) \times \{t\} \\
&\quad \text{stream of deleted elements} \\
\mathit{Rstream}(R) &= \bigcup_{t \in T} R(t) \times \{t\} \\
&\quad \text{stream of all elements}
\end{aligned}
$$

- In CQL *IStream* and *DStream* are syntactic sugar

Sensor Measurement CQL Example

Example

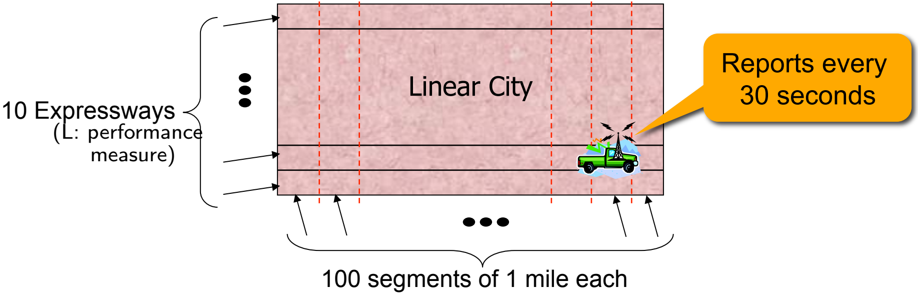
```
SELECT Rstream(m.sensorID)
FROM Msmt[Range 1] as m, Events[Range 2] as e
WHERE m.val > 30 AND
      e.category = Alarm AND
      m.sensorID = e.sensorID
```

- Stream join realized by join of window contents
- Output is a stream

Non-discrete Time Flows

- Taken literally, CQL window definitions work only for discrete flows of times
- Time flow: $(T, \leq) = (\mathbb{R}, \leq)$
- Input stream: $S = \{i\langle i\rangle \mid i \in \mathbb{N}\}$
- $\mathit{RStream}(S[\mathit{RANGE}\ 1\ \mathit{SLIDE}\ 1])$ is "stream" with cardinality of $\mathbb{R}$
- "Solution" in CQL hidden in stream engine layer
- Heartbeat with smallest possible time granularity

Linear City



Main Input Stream: Car Locations (CarLocStr)

car_id	speed	exp_way	lane	x_pos
1000	55	5	3 (Right)	12762
1035	30	1	0 (Ramp)	4539
...	...	...	...	...

Linear Road Benchmark

- 10 years old benchmark for stress testing relational DSMS
- Suite of continuous queries based on real traffic management proposals.
  - Stream car segments based on x-positions (easy)
  - Identify probable accidents (medium)
  - Compute toll whenever car enters segment (hard)
- Metric: Scale to as many expressways as possible without falling behind

  Lit: A. Arasu et al. Linear road: A stream data management benchmark. In M. A. Nascimento et al., editors, VLDB, pages 480–491. Morgan Kaufmann, 2004.

Toll Query

- Preconditions and Postconditions

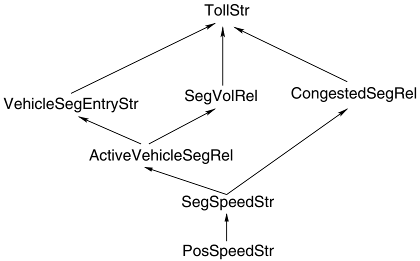
Trigger	Position report, q
Preconditions	q.Seg $\neq \overleftarrow{q}$.Seg, $l \neq$ EXIT
Output	(Type: 0, VID: v, Time: t, Emit: t', Spd: $Lav(M(t),x,s,d)$, Toll: $Toll(M(t),x,s,d)$)
Recipient	v
Response	t' - $t \leq$ 5 Sec

- $\boldsymbol{Toll(M(t),x,s,d) = 0}$ if in last 5 minutes either
  - congestion below 50 cars in the segment or
  - average speed $Lav(M(t),x,s,d)$ is below a given threshold or
  - segment is in vicinity of an accident
- else $\boldsymbol{Toll(M(t),x,s,d)} = 2 \times (\#(\boldsymbol{cars\ in}\ x, s, d) - 50)^2$

- Requires identification of **accidents**

Toll Query

- $\textit{\textbf{Toll}}(\boldsymbol{M(t),x,s,d}) = \mathbf{0}$ if in last 5 minutes either
  - congestion below 50 cars in the segment or
  - average speed $Lav(M(t),x,s,d)$ is below a given threshold or
  - segment is in vicinity of an accident
- else $\textit{\textbf{Toll}}(\boldsymbol{M(t),x,s,d}) = 2 \times (\#(\textit{\textbf{cars in}}\ x, s, d) - 50)^2$

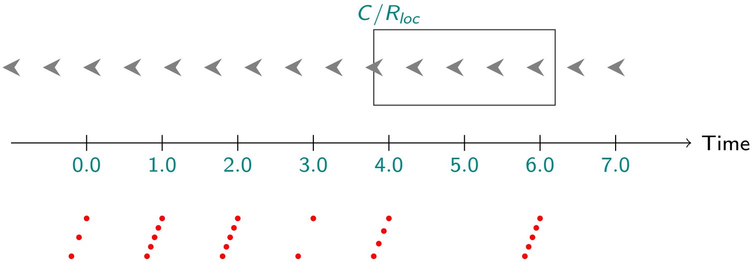


Query 6 `TollStr(vehicleId,toll)`: *This is the final output toll stream.*

```
Select Rstream(E.vehicleId,
               2 * (V.numVehicles-50)
                 * (V.numVehicles-50)
               as toll)
From VehicleSegEntryStr [Now] as E,
     CongestedSegRel as C,
     SegVolRel as V
Where E.segNo = C.segNo and
      C.segNo = V.segNo
```

High-Level Declarative Stream Processing

Local Reasoning Service



- Need to apply calculation/reasoning CR_{loc} locally, e.g.
  - arithmetics, timeseries analysis operations
  - SELECT querying, CONSTRUCT querying, abduction, revision, planning

High-Level and Declarative

- Declarative:
  Stream elements have "assertional status" and allow for symbolic processing

Example (Relational data streams)

Stream element $(sensor, val)\langle 3sec \rangle$ "asserts" that sensor shows some value at second 3

- High-Level:
  Streams are processed with respect to some background knowledge base such as a set of rules or an ontology.

Example (Streams of time-tagged ABox assertions)

Streams elements of form $val(sensor, val)\langle 3sec \rangle$ evaluated w.r.t. to an ontology containing, e.g., axiom $tempVal \sqsubseteq val$

High-Level and Declarative

- Declarative:
  Stream elements have "assertional status" and allow for symbolic processing

Example (Relational data streams)

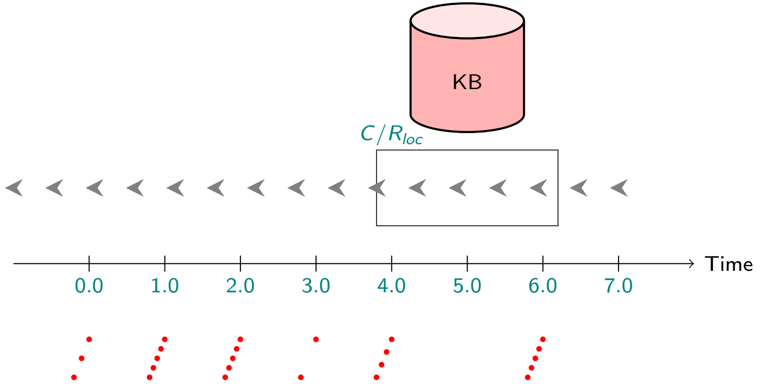
Stream element $(\mathit{sensor}, \mathit{val})\langle 3\mathit{sec}\rangle$ "asserts" that sensor shows some value at second 3

- High-Level:
  Streams are processed with respect to some background knowledge base such as a set of rules or an ontology.

Example (Streams of time-tagged ABox assertions)

Streams elements of form $\mathit{val}(\mathit{sensor}, \mathit{val})\langle 3\mathit{sec}\rangle$ evaluated w.r.t. to an ontology containing, e.g., axiom $\mathit{tempVal} \sqsubseteq \mathit{val}$

Local Reasoning Service



- Need to apply calculation/reasoning CR_{loc} locally, e.g.
  - arithmetics, timeseries analysis operations
  - SELECT querying, CONSTRUCT querying, abduction, revision, planning ($\Longrightarrow$ high-level & declarative)

Streamified OBDA

- Nearly ontology layer stream processing
  - CEP (Complex event processing)
  - EP-SPARQL/ETALIS, T-REX/ TESLA, Commonsens/ESPER
- RDF-ontology layer stream processing
  - C-SPARQL (della Valle et al. 09), CQELS
- Classical OBDA stream processing
  - $\text{SPARQL}_{\mathit{Stream}}$ (Calbimonte et al. 12) and MorphStream
- All approaches rely on CQL window semantics
- extend SPARQL or use some derivative of it
- Treat timestamped RDF triples but use reification

Example of Reified Handling

Example

```
SELECT ?windspeed ?tidespeed
FROM NAMED STREAM <http://swiss-experiment.ch/
                   data#WannengratSensors.srdf>
[NOW-10 MINUTES TO NOW-0 MINUTES]
WHERE
  ?WaveObs a ssn:Observation;
           ssn:observationResult ?windspeed;
           ssn:observedProperty sweetSpeed:WindSpeed.
  ?TideObs a ssn:Observation;
           ssn:observationResult ?tidespeed;
           ssn:observedProperty sweetSpeed:TideSpeed.
  FILTER (?tidespeed<?windspeed)
```

SRBench (Zhang et al. 2012)

- Benchmark for RDF/SPARQL Stream Engines
- Contains data from LinkedSensorData, GeoNames, DBPedia
- Mainly queries for functionality tests, with eye on SPARQL 1.1. functionalities

Example (Example Query (to test basic pattern matching))

Q1. Get the rainfall observed once in an hour.

- Tested on CQELS, $\text{SPARQL}_{\mathit{Stream}}$ and C-SPARQL
- Test results (for engine versions as of 2012)
  - Basic SPARQL features supported
  - SPARQL 1.1 features (property paths) rather not supported
  - Only C-SPARQL supports reasoning (on RDFS level) (tested subsumption and sameAs)
  - Combined treatment of static data plus streaming data only for CQELS and C-SPARQL

Language Comparison of SOTA Stream Engines

- Update in 2016
- We also mention Lübecks contribution STARQL (to be discussed in more detail in next lecture)

Name	Data Model	Union, Join Optional, Filter	IF	Aggregate	Property Paths	Time Windows	Triple Windows
Streaming SPARQL	RDF streams	Yes	No	No	No	Yes	Yes
C-SPARQL	RDF streams	Yes	Yes	Yes	Yes	Yes	Yes
CQELS	RDF streams	Yes	No	Yes	No	Yes	No
SPARQLStream	virt. RDF streams	Yes	Yes	Yes	Yes	Yes	No
EP-SPARQL	RDF streams	Yes	No	Yes	No	No	No
TEF-SPARQL	RDF streams	Yes	No	Yes	No	Yes	Yes
STARQL	virt. RDF streams	Yes	Yes	Yes	No	Yes	No

Name	W-to-S Op.	Cascading Streams	Intra window time	Sequencing	Pulse	Historic data
Streaming SPARQL	RStream	No	No	No	No	No
C-SPARQL	RStream	No	Yes	No	No	No
CQELS	RStream	No	No	No	No	No
SPARQLStream	R-,I-,D-Stream	No	Yes	No	No	No
EP-SPARQL	RStream	No	No	Yes	No	No
TEF-SPARQL	RStream	No	No	Yes	No	No
STARQL	RStream	Yes	Yes	Yes	Yes	Yes

Architecture Comparison of SOTA Stream Engines

Used Language	Input	Execution	Query Optimization	Stored Data	Reasoning
Streaming SPARQL	RDF streams	physical stream algebra	Static plan optimization	Yes	No
C-SPARQL	RDF streams	DSMS based evaluation with triple store	Static plan optimization	Internal triple store	RDF entailment
CQELS	RDF streams	RDF stream processor	Adaptive query processing operators	Stored linked data	No
SPARQLStream	Relational streams	external query processing	Static algebra optimizations host evaluator specific	Data source dependent	No
EP-SPARQL	RDF streams	logic programming backward chaining rules	No	No	RDFS, Prolog equivalent
TEF-SPARQL	RDF streams	Yes	No	Yes	Yes
STARQL	Relational streams	external query processing	Static algebra optimizations	Datasource dependent	Yes (DL-Lite$_{\mathcal{A}}$)

Lit: A. Bolles, M. Grawunder, and J. Jacobi. Streaming sparql - extending sparql to process data streams. In S. Bechhofer et al., editors, The Semantic Web: Research and Applications, vol. 5021 of LNCS, p. 448–462, 2008.
Lit: D. L. Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In L. Aroyo et al., editors, The Semantic Web - ISWC 2011, vol. 7031 LNCS, p. 370–388, 2011.
Lit: J.-P. Calbimonte, H. Jeung, O. Corcho, and K. Aberer. Enabling query technologies for the semantic sensor web. Int. J. Semant. Web Inf. Syst., 8(1):43–63, Jan. 2012.
Lit: D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in Etalis. Semantic Web, 3(4):397–407, 2012.
Lit: J.-U. Kietz, T. Scharrenbach, L. Fischer, M. K. Nguyen, and A. Bernstein. Tef-sparql: The ddis query- language for time annotated event and fact triple-streams. Technical Report IFI-2013.07, 2013.
Lit: Ö. Özçep, R. Möller, and C. Neuenstadt. A stream-temporal query language for ontology based data access. In KI 2014, vol. 8736 of LNCS, p. 183–194, 2014.

A stream reasoning community is forming

Everyone is interested in (high-level) stream processing now

- Various new stream reasoners (based on Datalog extensions)
- Stream reasoning + Machine Learning
- Stream reasoning + Verification
- Further benchmark ambitions and testing frameworks
- For recent progress see, e.g., 3rd stream reasoning workshop http://www.ifi.uzh.ch/en/ddis/events/streamreasoning2018.html