UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

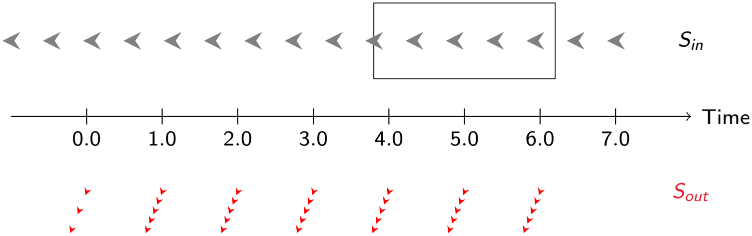
Stream Processing with Local Temporal Reasoning

Özgür L. Özçep

Workshop Stream Reasoning, Vienna
November 9, 2015

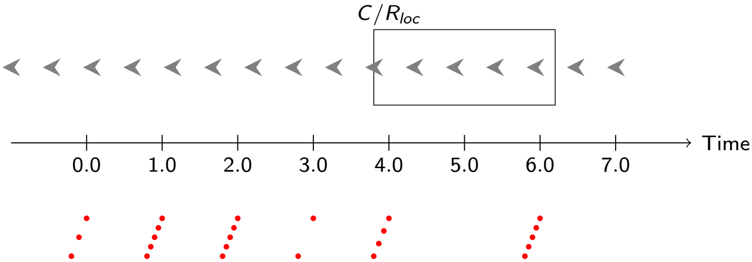
Local Reasoning on Streams

Taming the Potential Infinity of Streams



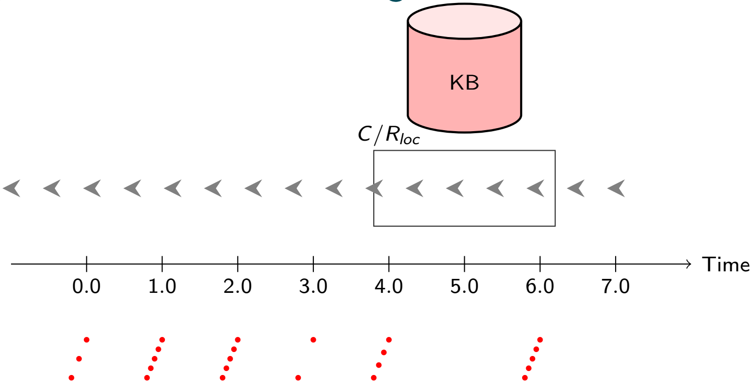
- Window operator as a means to cope with potential infinity
- Grab finite portion of stream and do something on it

Local Reasoning Service



- Local calculation/reasoning C/R_{loc}
  - arithmetics, timeseries-analysis operations
  - Entailment, satisfiability, query answering, abduction, revision, ...

High-Level Stream Processing



- Local calculation/reasoning C/R_{loc}
  - arithmetics, timeseries-analysis operations
  - Entailment, satisfiability, query answering, abduction, revision, ...
- Background knowledge KB: static data, historical data, learned data

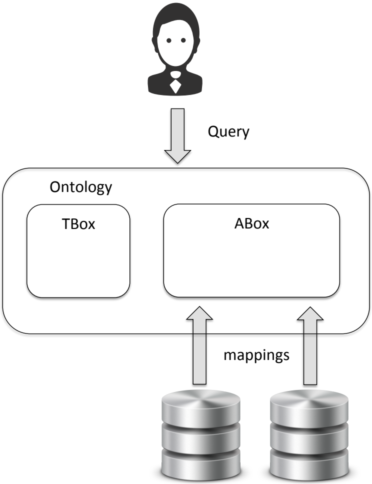
OBDA within OPTIQUE

OPTIQUE

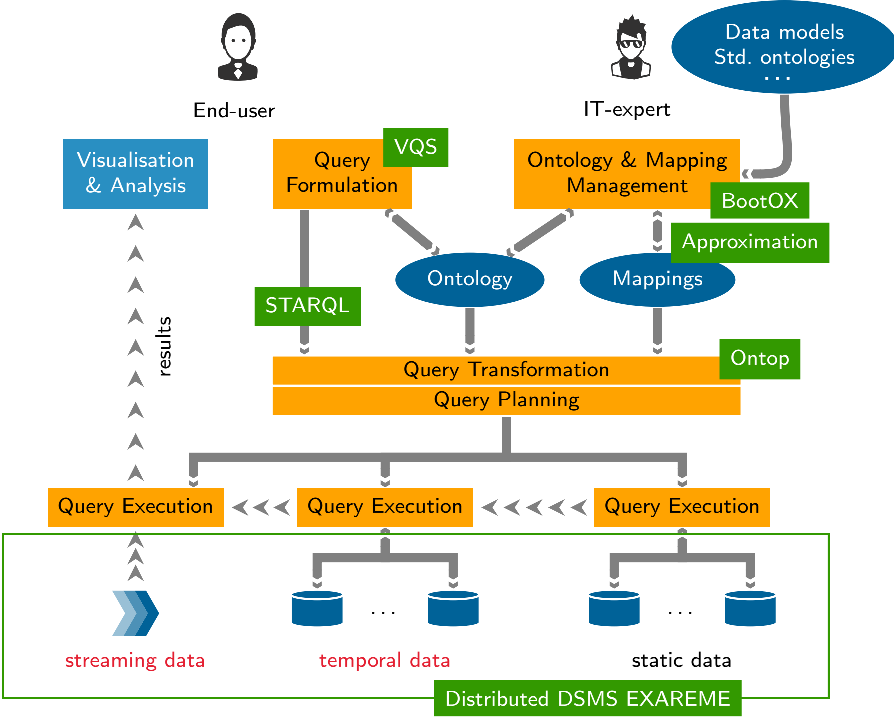
- EU 7th framework program (http://www.optique-project.eu/)
- Two big data use cases from industrial partners
  - STATOIL SAS: Querying data on wellbore related DBs
  - SIEMENS: Querying sensor and event data from (gas) turbines
- Cycle of constructing query, issueing it, and getting answers is bottleneck in both use cases
- Optique platform: OBDA with user support + optimizations on different levels + Streaming
- Lübeck (R. Möller, C. Neuenstadt, Ö.Ö.) responsible for stream-temporal OBDA module $\Longrightarrow$ STARQL

Ontology-Based Data Access

- Use ontologies as interface ...
- to access (here: query)
- data stored in some format ...
- using mappings

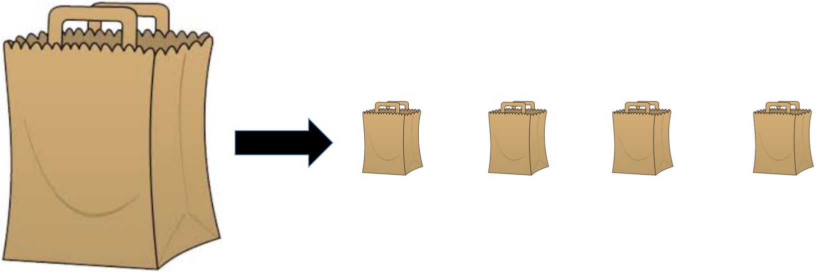


- Classical OBDA
  - Relational data
  - ABox is virtual
  - Query answering by rewriting/unfolding ("Reasoning by rewriting")
  - Weak ontology language (no qualified existentials on left-hand of inclusions)


Data models
Std. ontologies
...
End-user
IT-expert
Visualisation & Analysis
Query Formulation
VQS
Ontology & Mapping Management
BootOX
Approximation
Ontology
Mappings
STARQL
results
Query Transformation
Ontop
Query Planning
Query Execution
Query Execution
Query Execution
streaming data
temporal data
static data
Distributed DSMS EXAREME

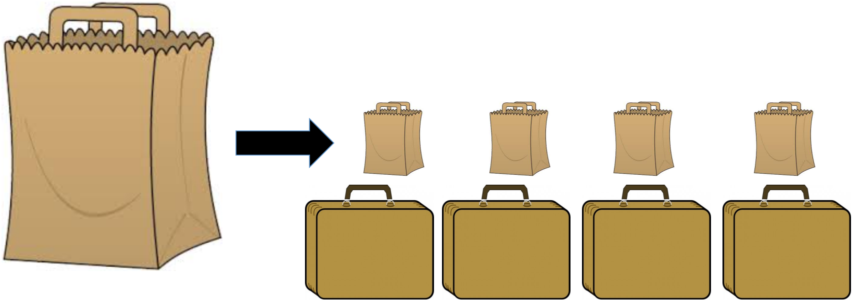
Reasoning within STARQL

Window Semantics in STARQL



- Group elements according to specified criterion (including timestamps) into mini-bags
- Technically: Result is a sequence of ABoxes/RDF graphs

Incorporating the Background Knowledge



Sequencing in STARQL

Information Need

Output every 1 minute those temperature sensors having value above 90 over the last minute

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in  [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Why at all Bother with State Sequences?

- Building microcosm for LTL like temporal reasoning on states
- But note
  - Temporal logic frameworks presuppose state sequences
  - In contrast, sequence construction is part of STARQL query

- Use case may require different types of states
  - cluster states using machine learning techniques
  - states corresponding to consistent ABoxes

Types of Reasoning in STARQL

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in  [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Types of Reasoning in STARQL

Determining **certain answers**

- Has to incorporate TBox (e.g. $BTTempSens \sqsubseteq TempSens$)
- Handled by rewriting

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in   [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Types of Reasoning in STARQL

Determining **certain answers**

- Has to incorporate TBox (e.g. $BTTempSens \sqsubseteq TempSens$)
- Handled by rewriting

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in  [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Types of Reasoning in STARQL

Local temporal reasoning on states

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in  [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Types of Reasoning in STARQL

Reasoning involved in **constructing the state sequence** (in particular for checking consistency of mini ABoxes)

Representation in STARQL

```
CREATE STREAM S_out AS
CONSTRUCT { ?sens rdf:type :tooHigh }<NOW>
FROM   S_in  [ NOW , NOW - 1 minute ]-> 1 minute,
       ABOX, TBOX
WHERE  { ?sens rdf:type TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL i IN seq FORALL ?x
    IF  { ?sens :hasVal ?x }<i> THEN ?x > 90
```

Theoretical Results

- **State elimination**
  - State abstraction means additional layer in OBDA stack
  - Nonetheless, it can be eliminated
    (Ö., Möller, Neuenstadt 2014, 2015)
- **Relation to LTL approaches**
  - Backend systems mostly have domain independent languages
  - LTL like query languages not domain independent
  - TCQs: CQs combined with LTL (Borgwardt et al. 13)
  - A fragment of STARQL embeds a safe fragment of TCQs
    (Ö., Möller, Neuenstadt 2015)

Theoretical Results

- **State elimination**
  - State abstraction means additional layer in OBDA stack
  - Nonetheless, it can be eliminated (Ö., Möller, Neuenstadt 2014, 2015)
- **Relation to LTL approaches**
  - Backend systems mostly have domain independent languages
  - LTL like query languages not domain independent
  - TCQs: CQs combined with LTL (Borgwardt et al. 13)
  - A fragment of STARQL embeds a safe fragment of TCQs (Ö., Möller, Neuenstadt 2015)

Practical Results

- Implemented STARQL sub-module with optimizations
- Transformation realized to backend EXAREME
- Optimizations for distributed stream processing in EXAREME

- Multiple Query/multiple stream handling
  - Monitor different components (turbines, sensors)
  - Monitor different hand-crafted well-proven patterns

- Specific statistical and time-series operators
  - Pearson-correlation (e.g. for detecting out faulty sensors)
  - Calls for specific optimizations (local-sensitive hashing)

Future Work

Stream Reasoning for NLP

- Intention: Use stream semantics and techniques for natural language processing (NLP)
- One of the (very few) application scenarios where stream-processing historical data makes sense
  - You could read a text in "parallel" but here, "order really matters":
  - Meaning of sentence depends on meanings of preceding sentences
- Discoure representation theory (DRT): Capture super-sentence meaning by discourse structures
- Calls for state-based stream processing with a **scopus** storing discourse structures

Challenges

- Need for state-based stream processing with a **scopus** storing discourse structures
  - Discourse structure dynamically updated
  - In general may grow arbitrarily

- Need for abduction style reasoning (Sherlock Holmes style reasoning)
  - From observations to possible explanations
  - Have to constrain search space, anytime abduction

- Different orders to incorporate
  - sentence (arrival) ordering (so)
  - causal ordering (co)
  - temporal ordering(s) (to)

Example

- Bob cried. Alice consoled him. (so corresponds to to)
- Bob cried. Alice insulted him. (so corresponds to co)

Thank you for your attention!