Semantics for OBDA on Streaming Data Sources

Özgür L. Özcep
Hamburg University of Technology

1. Motivation: The Siemens use case

Need for stream processing: The Siemens use case in OPTIQUE

- Service centers for thousands of geographically distributed gas/steam turbines

-> Need to handle data from multiple sources

- Time-stamped sensor data of several TB

-> Need to handle **acquisitional streams** (regular)

- Event data ("alarm triggered at time T") of several GB

-> Need to handle **event streams** (irregular)

Example queries

- Determine all locations of turbines containing a component X in version 3.2, and having not been serviced within the last 6 months.
- Determine all turbines for which in the past 5 maintenance activities action A followed by action B was carried out, but not action C.
- Determine all turbines T with an error “ERROR_Y“

  Errory_Y defined by at least five percent decrease of measured value M_K of component K followed by a statistically significant increase of measured value M_T of T

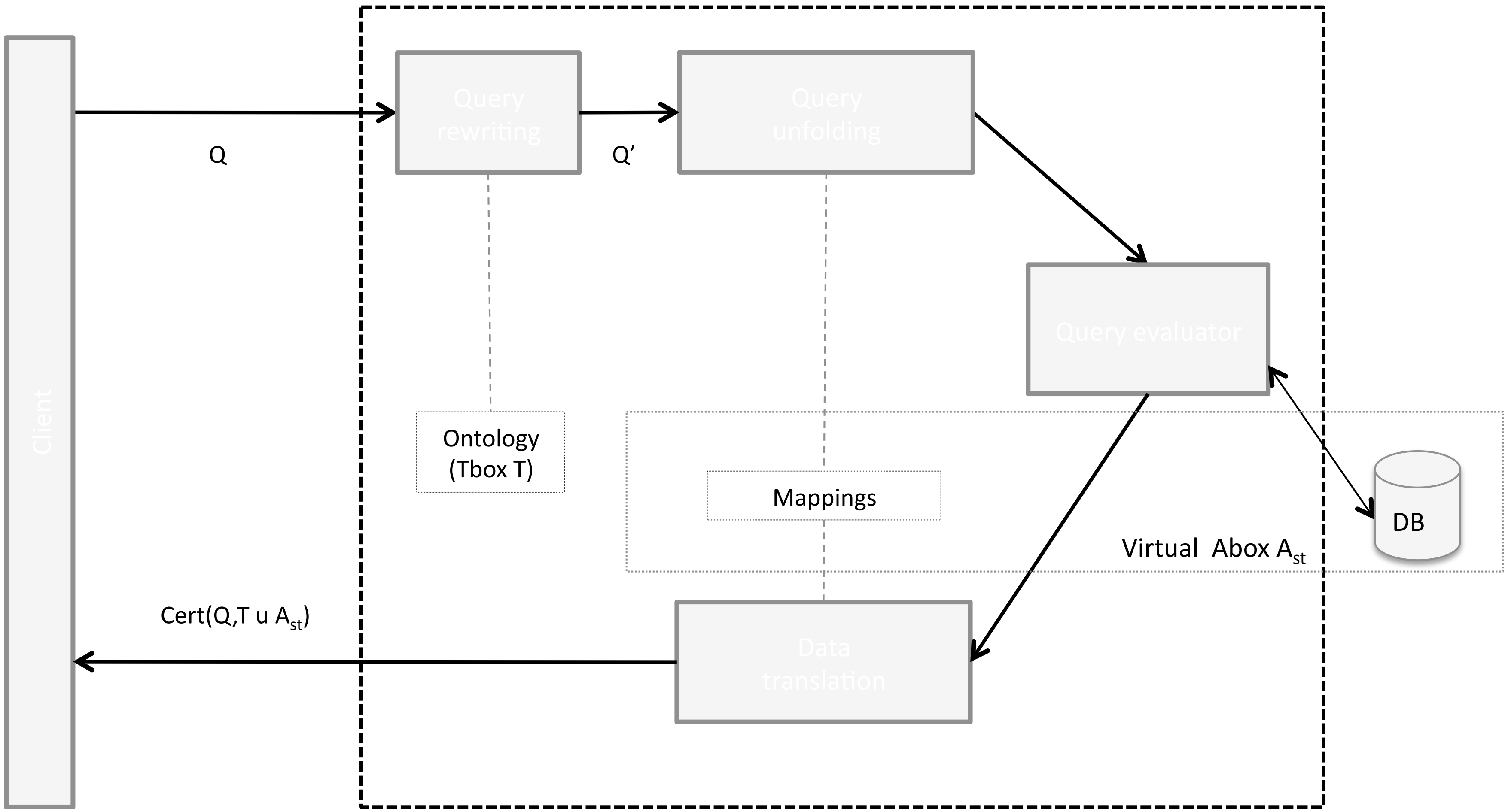
Ontologies and mappings

- Ontology as common model for multiple sources
  - E.g. concept of measure, sensor, reliability
- Ontology for describing structure of powerplants
- Mappings for grounding

- General OBDA model has to be extended
  - Provide mappings between ontology and streams
  - Provide query language with time/window semantics

2. Streamifying ontology based data access

OBDA with query rewriting for relational DBs

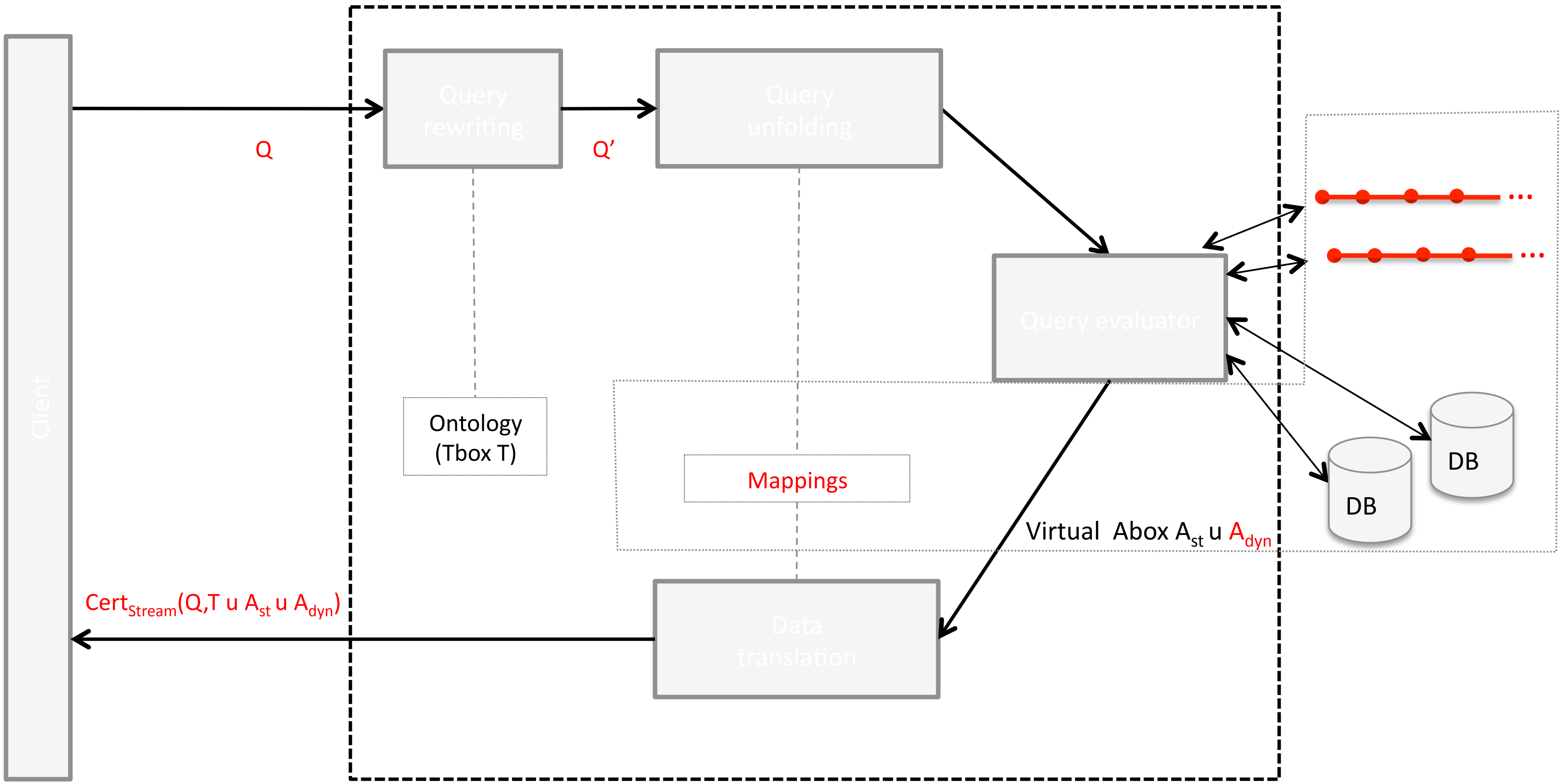


Weakly streamifying OBDA

- Desiderata
  - Queries that are sufficiently expressive for use case
  - Keep computational feasibility of lightweight DLs
- No explicit time or stream constructors in ontology (Tbox)
  - Highly complex logics would result even for DL-Lite (Artale et al. 2010)

  -> use DL-Lite ontology
- Grounding with extended mappings for streams
- Time/Stream constructors in query language
  - Extend conjunctive queries with time/stream constructors
  - that are `simple’

OBDA with query rewriting for relational DBs and streams

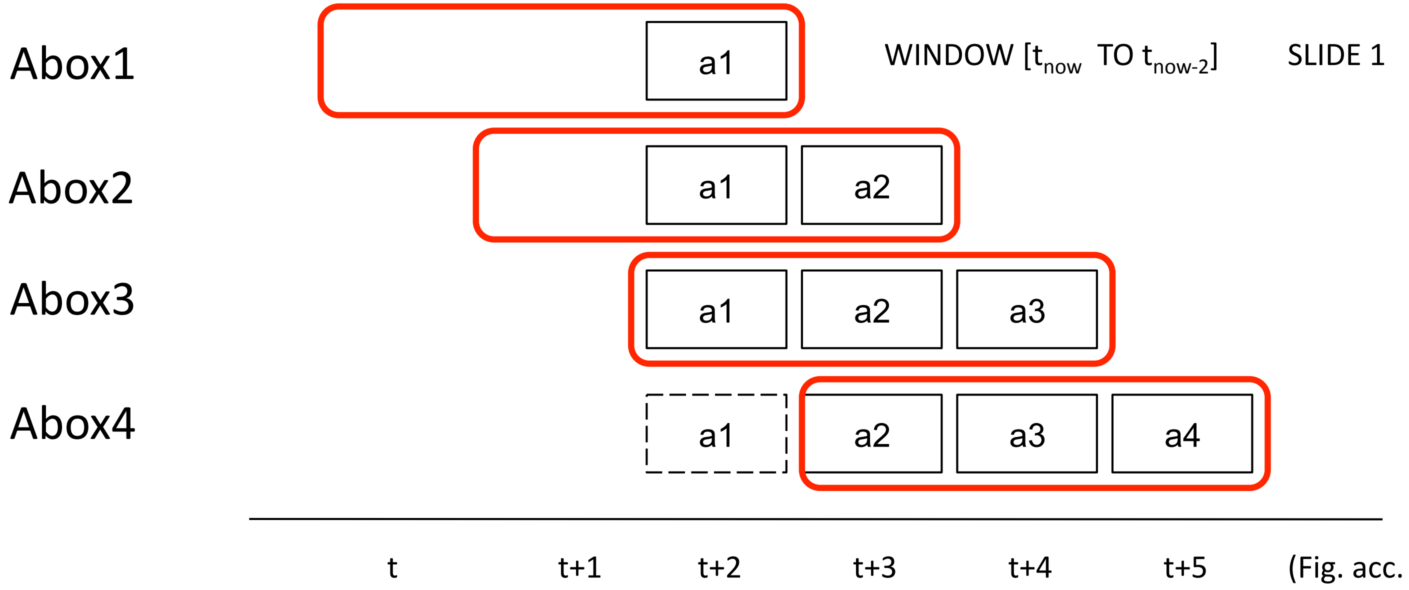


3. Queries on streams

Streamifying OBDA: Extending the query language

- Different Approaches
  - EP-SPARQL (Arasu/Widom, 2005)
  - CQL (Arasu et al. , 2006)
  - StreamingSPARQL (Bolles et al., 2008)
  - C-SPARQL (Barbieri et al., 2010)
  - $\text{SPARQL}_{\text{Stream}}$ (Calbimonte et al., 2010)
  - CQELS (Le-Phuoc et al., 2011)
- Common feature: “Snapshot” semantics based on sliding windows

Streams and windows

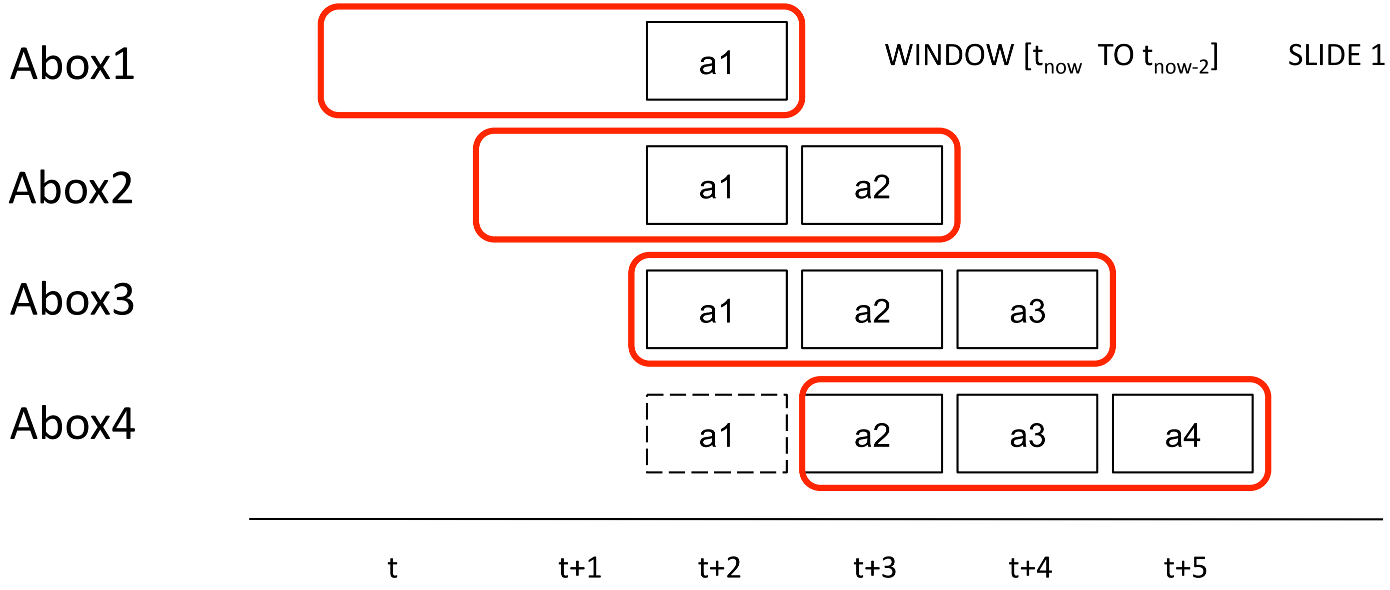


(Fig. acc. to Calbimonte et al., 2010)

+ Stream: (Infinite) set of time stamped data (Abox assertions)
+ Window: Streams x Timestamps -> Aboxes

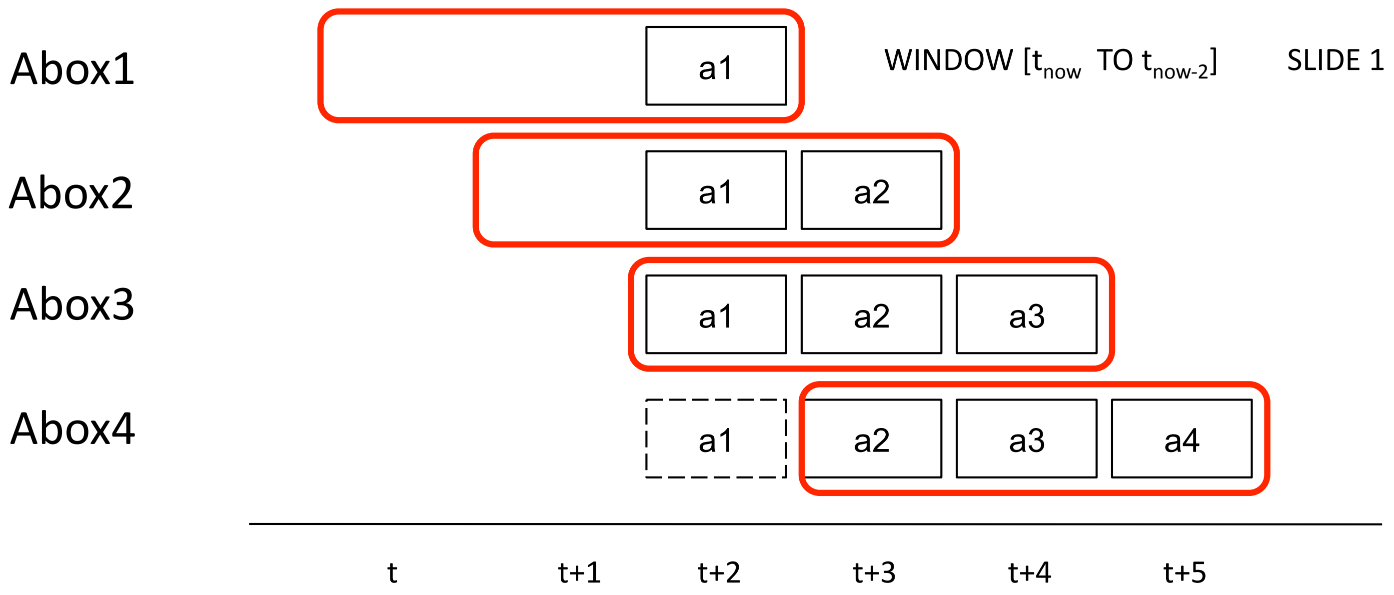
  Streams -> Abox stream

Example



- $Q\_[t_{now},t_{now}-2,1]$ (x,y) = Turbine(x) and measureVal(x,y)$[t_{now},t_{now}-2,1]$
- $Cert_{Stream}(Q\_[t_{now},t_{now}-2,1]$ (x,y) , T u A_{st} u A_{dyn}) = ?
- Time and streaming not relevant for rewriting; classical evaluation over Aboxes

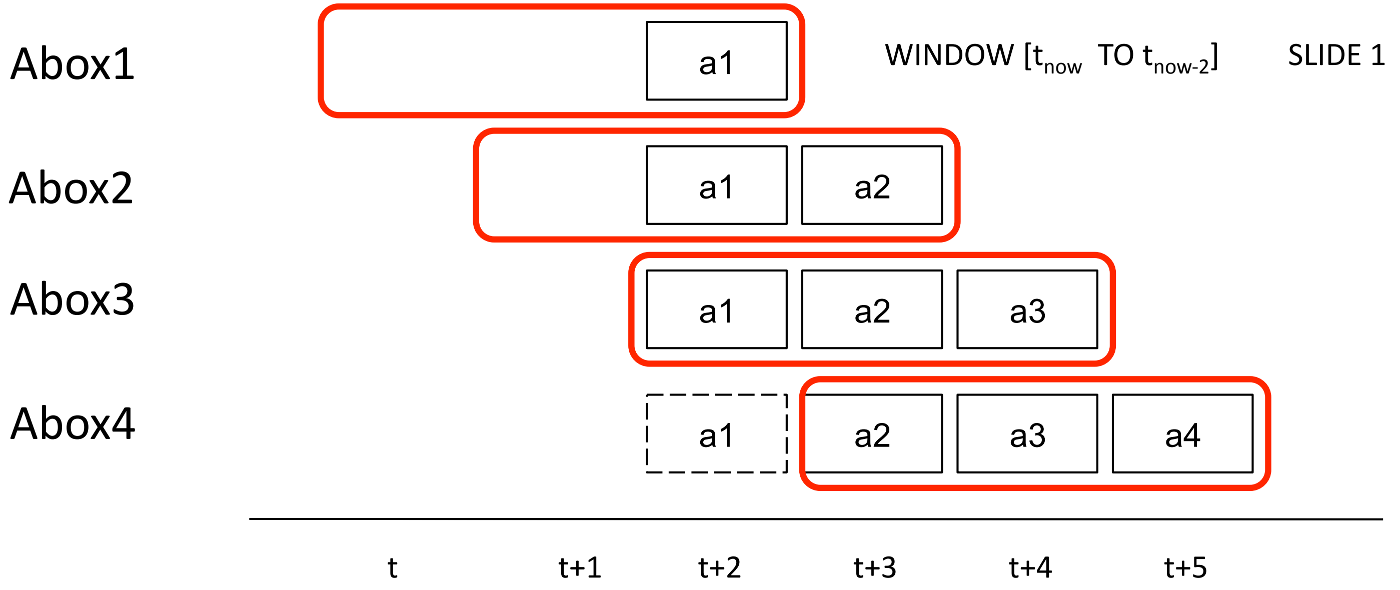
Example (continued)



- $Q\_[t_{now},t_{now}-2,1]$ (x,y) = Turbine(x) and measureVal(x,y)$[t_{now},t_{now}-2,1]$
- Q': Rewritten query without window
- $Cert_{Stream}(Q\_[t_{now},t_{now}-2,1]$ (x,y) , T u A_{st} u A_{dyn}) =

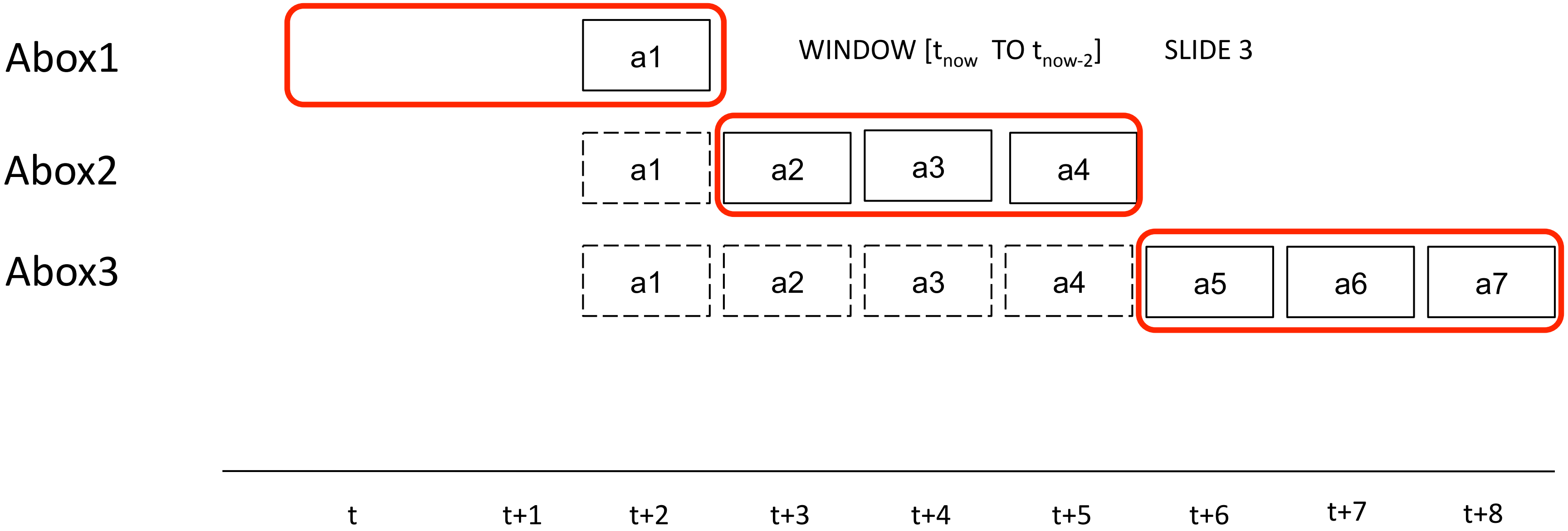
  {Cert(Q', A_{st} u Abox1), Cert(Q',A_{st} u ABox2), …}

Overlapping



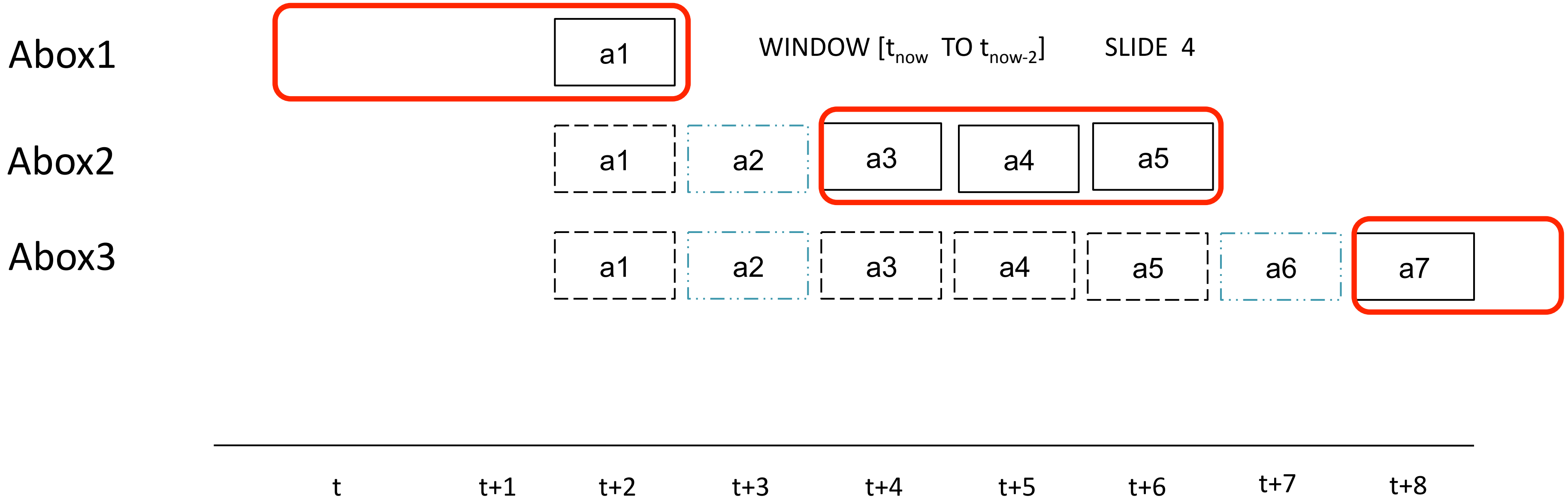
window size (= 3) > slide size (=1)

Covering



window size (= 3) = slide size (=3)

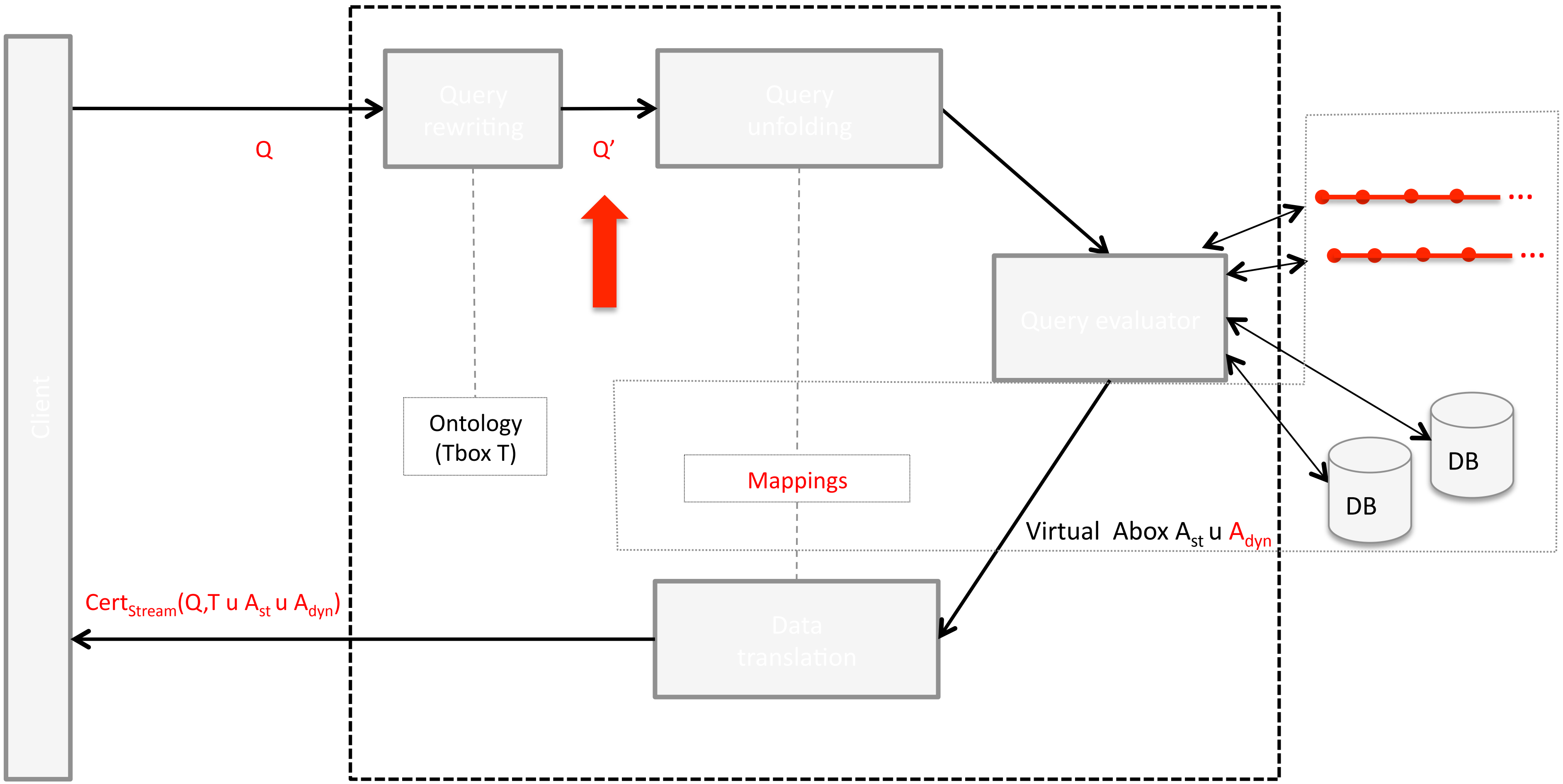
Sampling



Window size (= 3) < slide size (=4)

4. Target language for rewriting

OBDA with query rewriting for relational DBs and streams



Target language for query rewriting

- FOL rewritability: Cert(Q, T u A_{st}) = ans(Q', DB(A_{st}))
  - DB(A_{st}): Herbrand model of A_{st}
  - Q' is in FOL
  - Answering FOL queries in data complexity (ignore size of query) AC^0
- Last use case query not in pure FOL
  - may need counting , aggregation
  - over potentially infinite sequences
- FOL^{ext} rewritability: Cert(Q, T u A_{st} u A_{dyn}) = ans(Q',DB (A_{st} u A_{dyn}))
  - Q' in FOL^{ext}
  - data complexity

Target language for query rewriting

- Non-recursive SQL corresponds to FOL + Counting + Aggregation
- Data complexity not AC^0 but in TC^0 (Libkin, 2003)

  (Parity problem in $TC^0 \backslash AC^o$)
- TC^0 : Boolean circuits allow for threshhold gates
- AC^0 proper subset of TC^0. TC^0 still not distinguished from NP

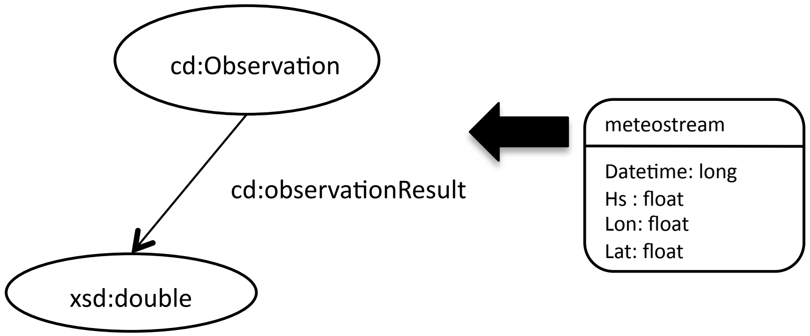
5. Black box and white box approaches

Black box approach (Calbimonte et al., 2010)

- Black box approach for OPTIQUE prototype implementation
- Mappings: S2O = Streamified version of R2O (Barrasa et al., 2004)
- Target unfolding language: Sneeql = Streamified version of SQL
- Query language: $\text{SPARQL}_{\text{Stream}}$
- Window-to-stream operators

S2O

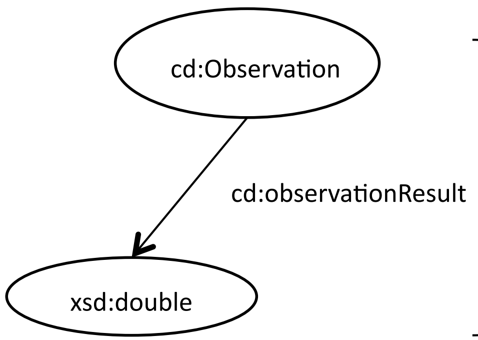
```
<conceptmap-def id="Observation_wind"
name="http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#Observation"
virtualStream="http://www.semsorgrid4env/ccometeo.srdf">
 <uri-as>
  <operation oper-id="concat">
     <arg-restriction on-param="string1">
       <has-value>http://www.semsorgrid4env.eu/data#ObservationWind</has-value>
    </arg-restriction>
    <arg-restriction on-param="string2">
       <has-column>meteostream.DateTime</has-column>
    </arg-restriction>
  </operation>
 </uri-as>
```




```
<attributemap-def
name="http://www.semsorgrid4env.eu/
          ontologies/CoastalDefences.owl#observationResult"
dataType="xsd:double">
…
</attributemap-def>
```

Example SPARQL$_{Stream}$

“Provide me with the wind speed observations over the last minute in the Solent Region”

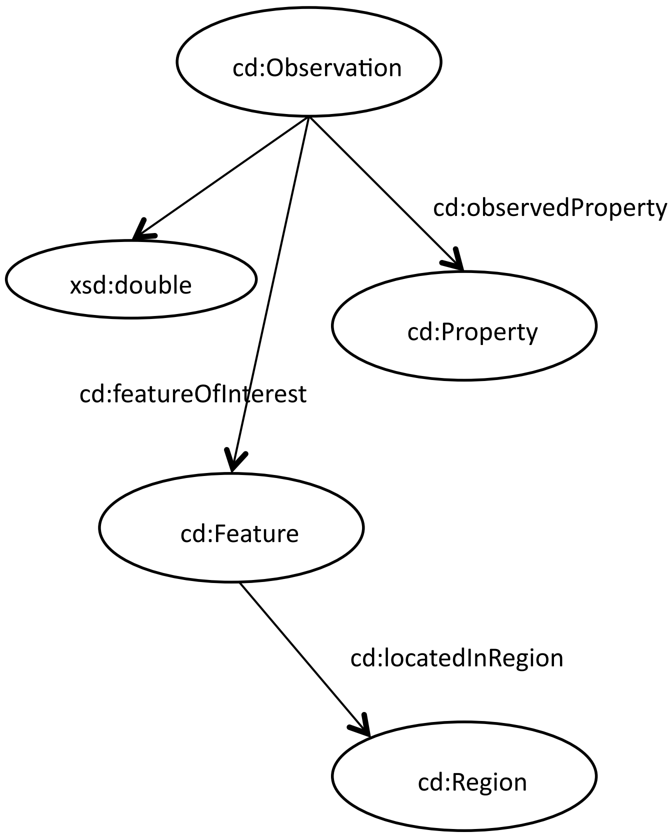


STREAM <*http://www.semsorgrid4env.eu/ccometeo.srdf*>

...
...
(<ssg4e:Obs1,rdf:type, cd:Observation>, t_i),
(<ssg4e:Obs1,cd:observationResult,”34.5”>, t_i),
(<ssg4e:Obs2,rdf:type, cd:Observation>, t_{i+1}),
(<ssg4e:Obs2,cd:observationResult,”20.3”>, t_{i+1}),
...
...

Example SPARQL$_{\text{Stream}}$

“Provide me with the wind speed observations over the last minute in the Solent Region”




```
PREFIX cd: <http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#>
PREFIX sb: <http://www.w3.org/2009/SSN-XG/Ontologies/SensorBasis.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?windspeed ?windts
FROM STREAM <http://www.semsorgrid4env.eu/ccometeo.srdf>
[ NOW – 1 MINUTE TO NOW – 0 MINUTES ]
WHERE
{
 ?WindObs a cd:Observation;
   cd:observationResult ?windspeed;
   cd:observationResultTime ?windts;
   cd:observedProperty ?windProperty;
   cd:featureOfInterest ?windFeature.
 ?windFeature a cd:Feature;
   cd:locatedInRegion cd:SolentCCO.
 ?windProperty a cd:WindSpeed.
 }
```

Window-To-Stream Operators

- Stream of windows to stream of time tagged assertions

- IStream: Newly inserted assertions w.r.t. preceding window
- DStream: Deleted assertions w.r.t. to preceding window
- RStream: All assertions in window

White box approach

- Use for extended/optimized version for OPTIQUE
- Direct control on query evaluation
- Optimization by dynamic adaptation
- See e.g. (Le-Phuoc et al., 2011)

6. Conclusion

Conclusion

- Window semantics to capture expressive queries
- Clarify the semantics of implemented stream query languages by mapping into some extension of FOL
- Theoretical work on FOL^{ext} rewriting
- Use black box approach for prototype version as Calbimonte et al. 2010
- White box approach (Le-Phuoc et al., 2011) for optimized version
- Predictive Diagnostics
  - Generate dynamic stream of hypotheses on expected values which may change depending on observation streams