Stream Reasoning – Research and Use cases at the Institute of Information Business

Axel Polleres & Javier Fernandez, WU Wien

Outline

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Disclaimer: **NO concrete work on Stream Reasoning in my group at the moment**, but still on 3 related topics...

1. Dynamic Linked Data & Open Data
2. Efficient RDF Stream Interchange (ERI)
3. SPARQL 1.1 Update & Entailment as a basis for Stream Reasoning Semantics?

1. Dynamic Linked Data & Open Data:

Collecting changes in Linked and Open Data:

- Dynamic Linked Data Observatory
- Open Data Portalwatch
- The DBPedia Wayback machine

1.1 Dynamic Linked Data Observatory

Started in DERI since 2012, **weekly snapshots** of crawla of Linked Data... Interesting dataset for investigating LOD dynamics.

http://dyldo.deri.org/

Publications:

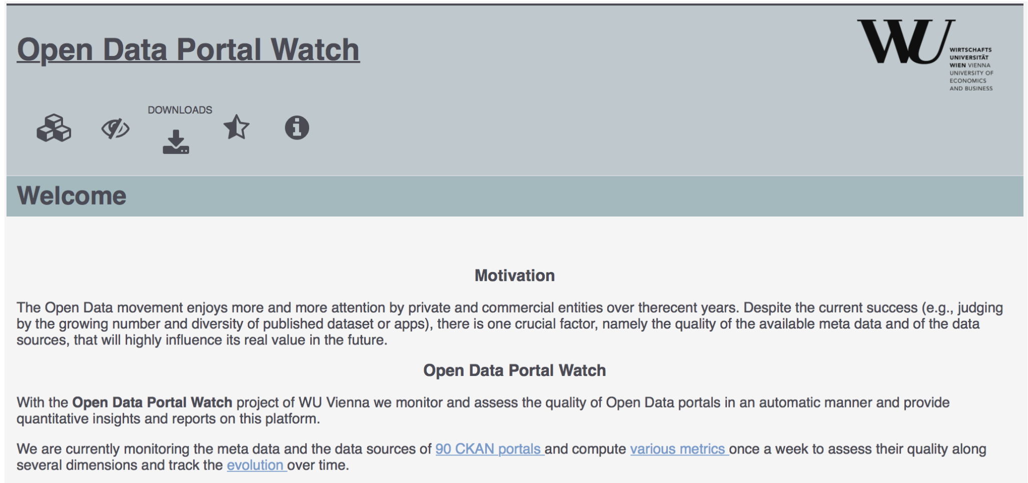
Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O'Byrne, Aidan Hogan."***Observing Linked Data Dynamics***". In the Proceedings of the 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, 26–30 May, 2013.

Tobias Käfer, Jürgen Umbrich, Aidan Hogan and Axel Polleres, "***Towards a Dynamic Linked Data Observatory***", In the Proceedings of the Linked Data on the Web WWW2012 Workshop (LDOW 2012), Lyon, France, 16 April, 2012.

1.2 Open Data Portal Watch

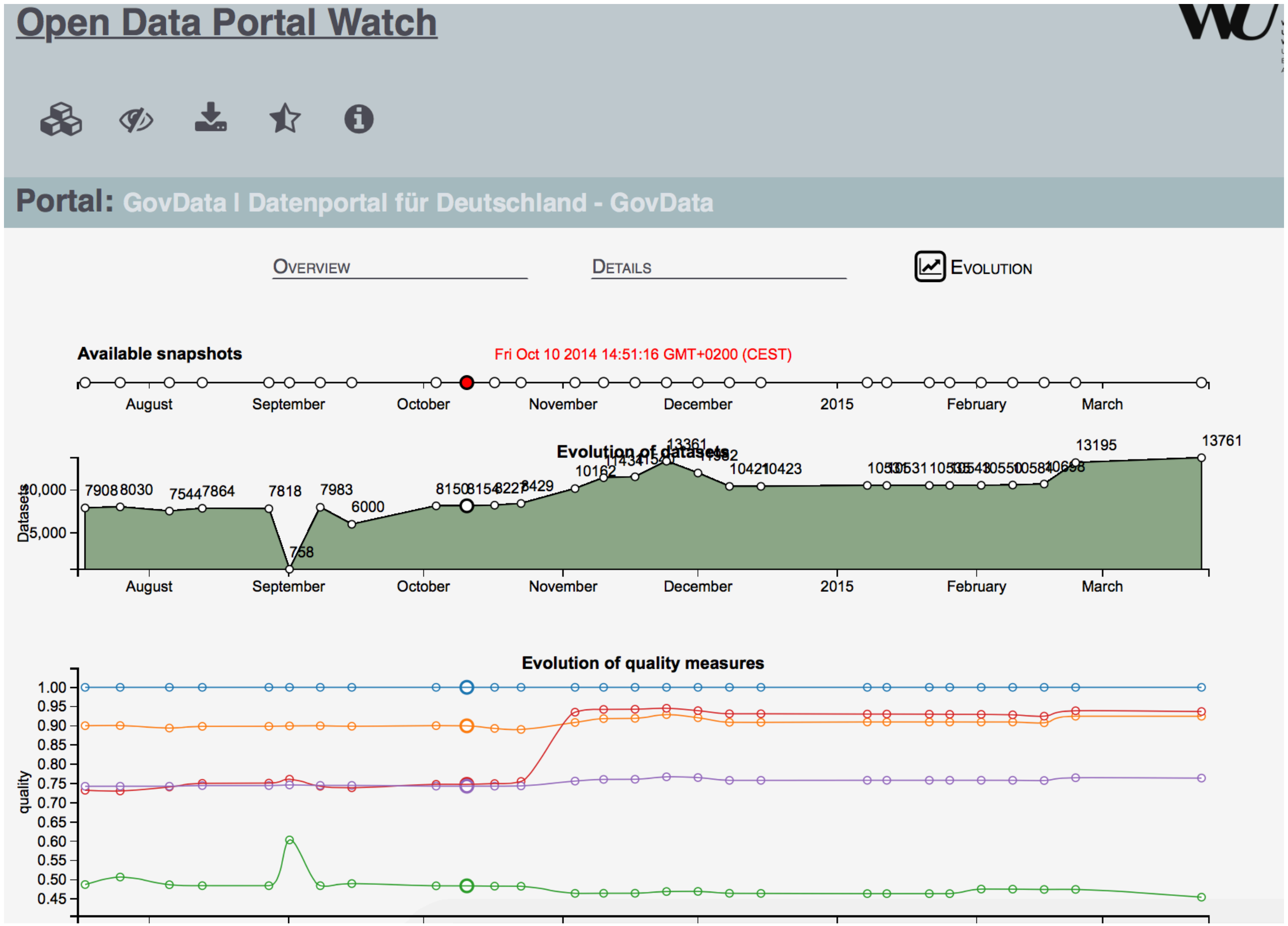
http://data.wu.ac.at/portalwatch/

- Started in 2014 – similar idea, different dataset:
  - Weekly crawls of Data and Metadata from over 90 Open Data Portals (CKAN, Socrata, etc)
- Goal: Quality assessment
- Evolution tracking
  - Meta data
  - Data

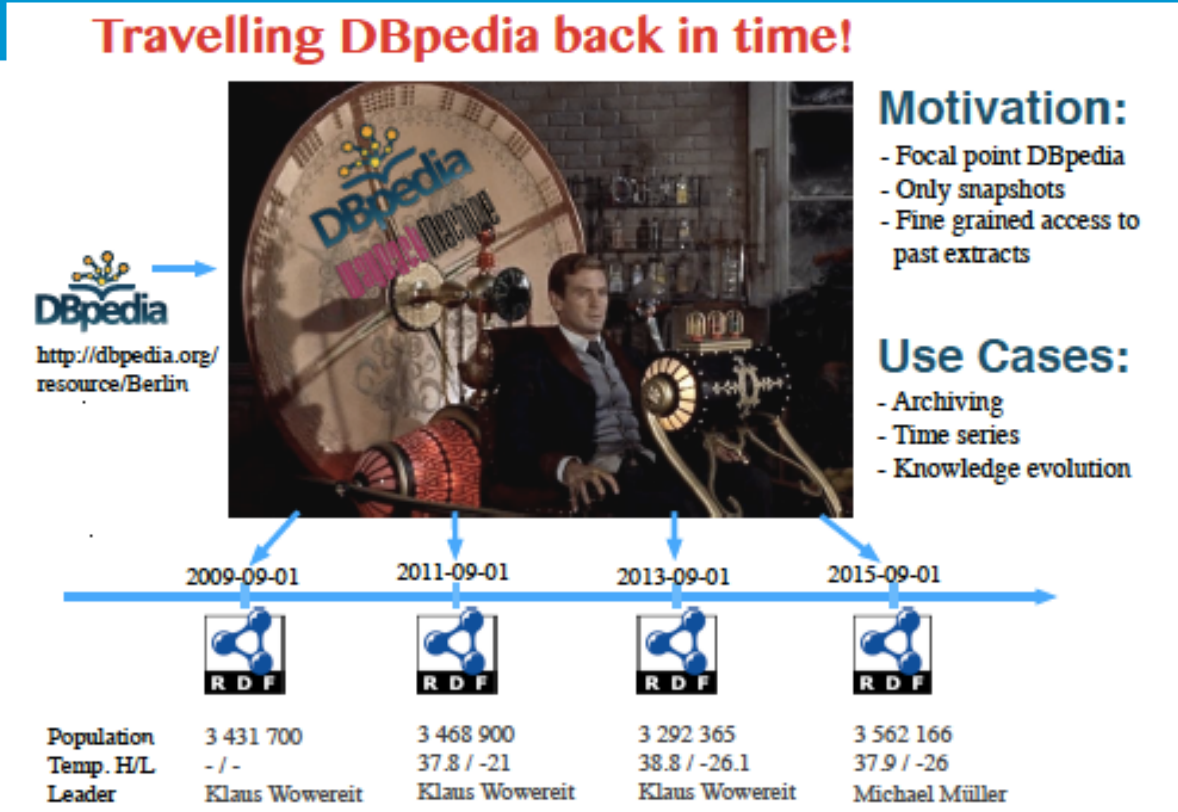


  - Jürgen Umbrich, Sebastian Neumaier, and Axel Polleres. Quality assessment & evolution of open data portals. In *IEEE International Conference on Open and Big Data*, Rome, Italy, August 2015. **Best paper award**.

1.2 Open Data Portal Watch



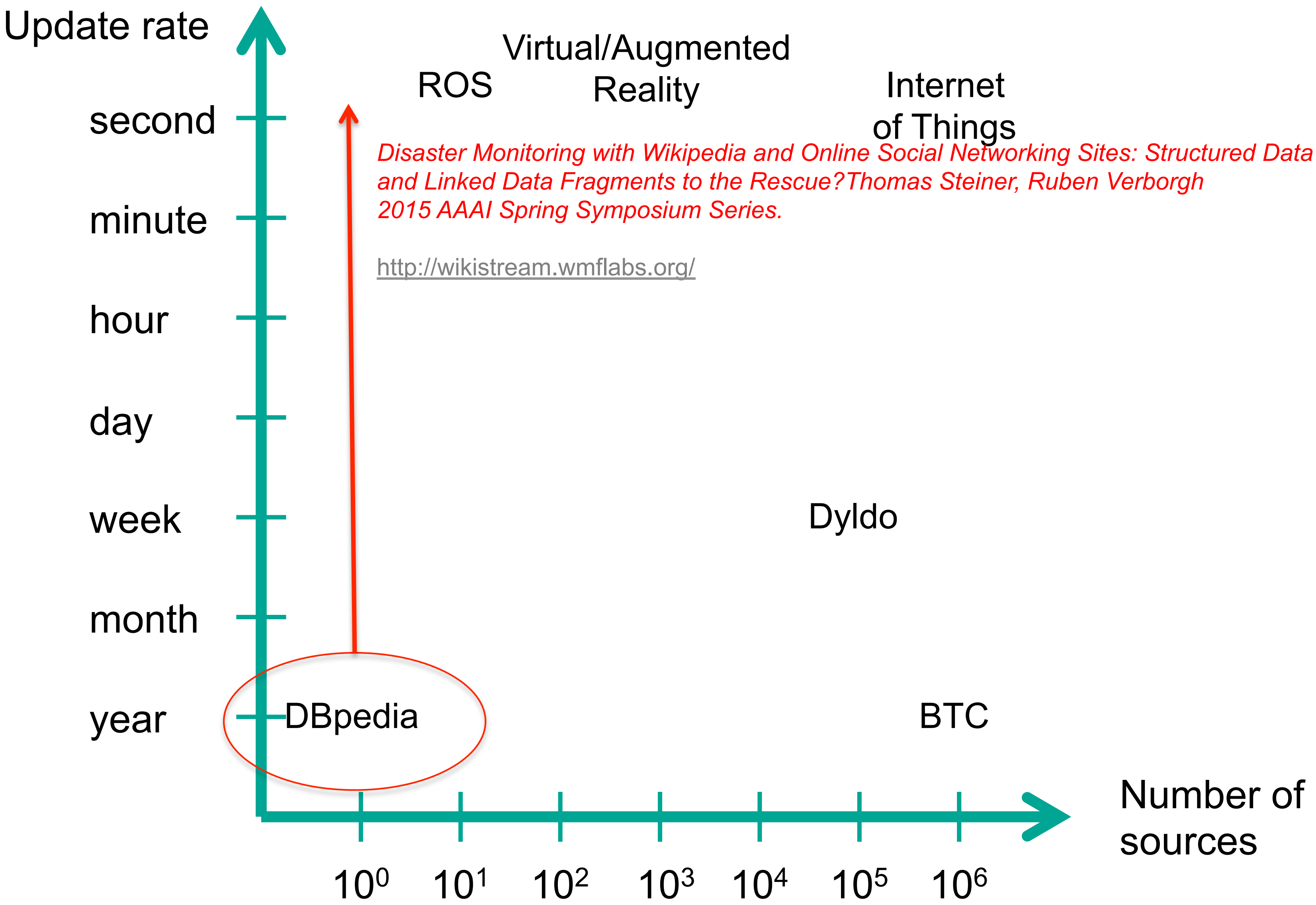
Dbpedia Wayback Machine



http://data.wu.ac.at/wayback/ ... Different concept: create historical RDF Data from wikipedia/dbpedia revision history

"The DBpedia wayback machine" by Javier Fernandez, Patrik Schneider, Jürgen Umbrich, September 2015, SEMANTiCS2015 (Best Poster).

Motivation: Data Integration and System Interoperation at Scale



1. Dynamic Linked Data & Open Data – Discussion:

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

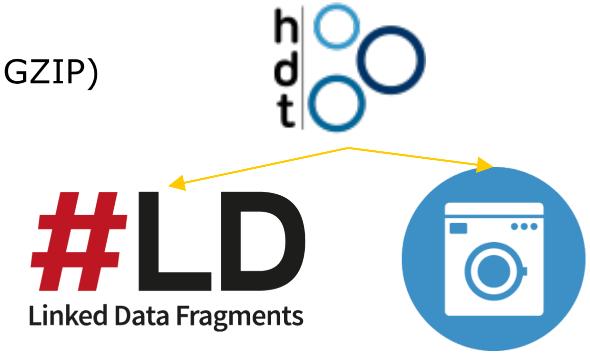
Stream Reasoning related questions:

- How can we efficiently store and query archival data?
- Where does reasoning help us? (e.g. detect anomalies, conflicts, assume defaults in case of missing crawls, etc.)

- Javier D. Fernández, Axel Polleres, and Jürgen Umbrich. Towards efficient archiving of dynamic linked open data. In *Managing the Evolution and Preservation of the Data Web - First Diachron Workshop at ESWC 2015*, pages 34-49, Portorož, Slovenia, May 2015

2. Efficient RDF Stream Interchange (ERI)

- Lightweight **Binary RDF (HDT)**
  - Highly **compact** serialization of RDF (slightly bigger than GZIP)
  - compact RDF store (without prior decompression)
  - Basics for successfully projects
    - Linked Data fragments
    - LOD Laundromat



- **RDF Data Streams**

Querying & Efficient Serialization of streams of LD

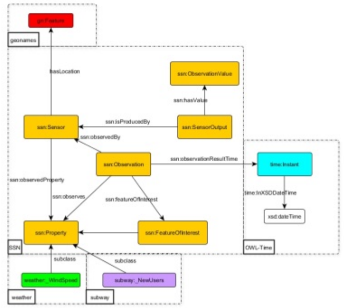
- In the light of IoT/Industry 4.0, etc. ...

- We are monitoring the W3C RSP (RDF Stream Processing) CG, but also EXI (Efficient XML Interchange and WoT WG

Efficient Serialization of streams of LD

	Plain: Turtle/ Trig/ JSON-LD	Plain +Compression (e.g. gzip)	HDT	Streaming HDT	RDSZ	RDF/XML + EXI	ERI
Streamable	Yes	Yes	No	Yes	Yes	Yes	Yes
Scalable	Limited	Yes	Yes	No	Yes	Yes	Yes
Easy (fast) to create and parse	Yes	Limited	Limited	Yes	Limited	Limited	Yes
Compact	No	Yes	Yes	Limited	Yes	Yes	Yes
Parametrizable: compression/time	No	Limited	Yes	No	Limited	Limited	Yes

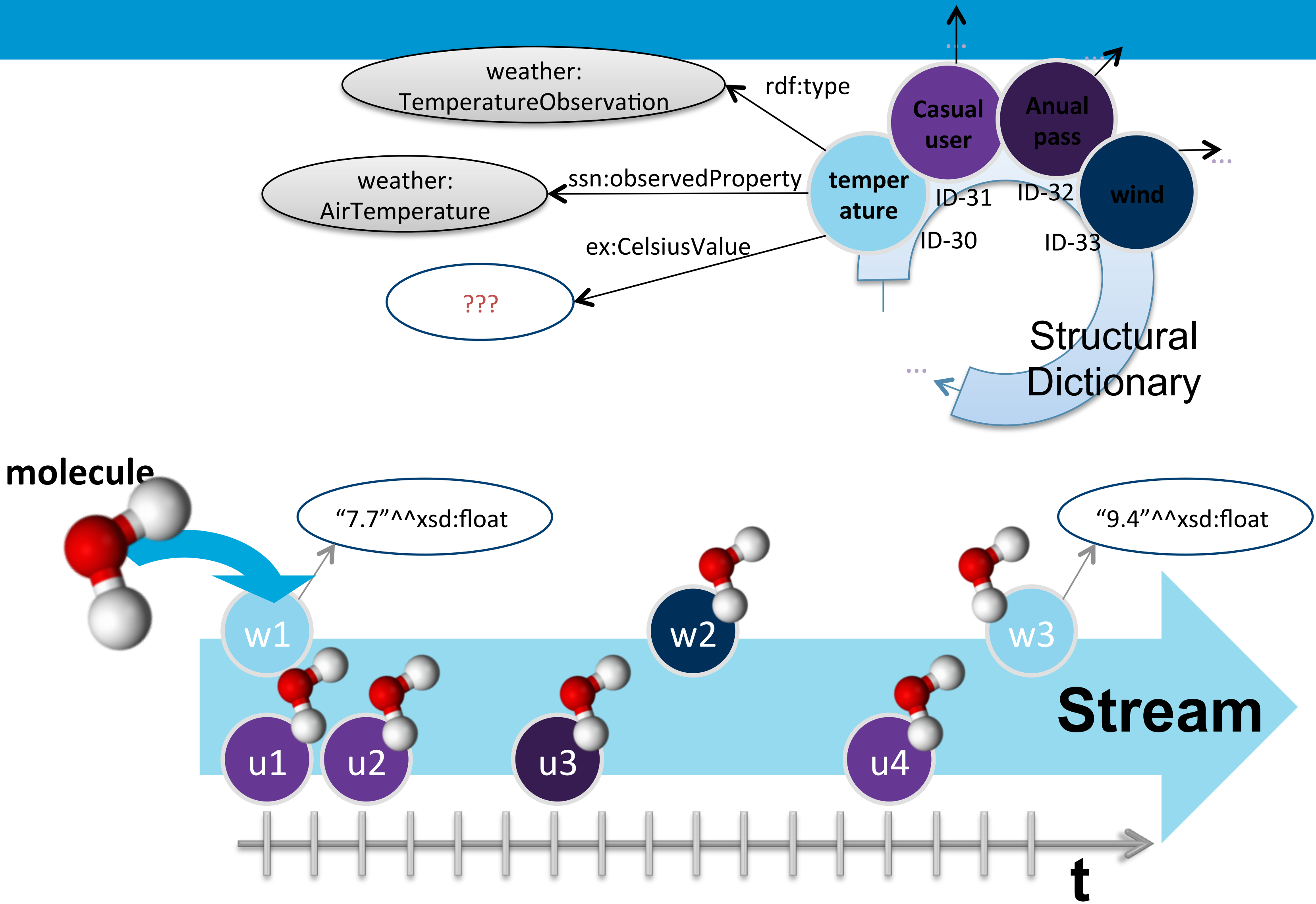
Efficient RDF Interchange (ERI) Format – Basic Concepts



- (Assumption) Most RDF streams are **well structured**
  - the **structure** is well-known by the data provider
  - the number of **variations** in the structure are limited

- Efficient RDF Interchange (ERI) Format encodes the information at two levels:
  - A **sliding** dictionary of **structures:** Structural Dictionary
  - The **concrete value** for each predicate

EFFICIENT RDF INTERCHANGE (ERI) FORMAT – Basic Concepts



2. Efficient RDF Stream Interchange (ERI)– Discussion:

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Stream Reasoning related questions:

- Where does Reasoning help us? (e.g. compact representation by stripping off implicit information)

- Vague idea what could be related here:

Reinhard Pichler, Axel Polleres, Sebastian Skritek, and Stefan Woltran. Complexity of redundancy detection on rdf graphs in the presence of rules, constraints, and queries. Semantic Web - Interoperability, Usability, Applicability (SWJ), 4(4), 2013.

3. SPARQL 1.1 Update & Entailment as a basis for Stream Reasoning Semantics?

SPARQL 1.1 can update a stream and do entailment regimes, however...

Standardization of SPARQL 1.1 Update, and SPARQL 1.1 Entailment Regimes with triple stores implementing those standards (rewriting- or materialization-based) **is silent about their interaction**

What's been done already?

- What do off-the-shelf triple stores do?
  - **Entailment** typically handled
    - at (bulk) loading *by* ***materialization*** ***, or***
    - at *query time by* ***rewriting***

    but **not in the context of Updates**.
  - no "standard" behavior for **Delete** & **Insert** upon materialized stores.
  - interplay of Entailments and Update left out in the SPARQL 1.1 spec.
- What does the literature say?

  Approaches *in the literature on updates and RDFS* (or also DLs) limited to ***atomic update*** operations…
  - [Gutierrez et al., ESWC2011] ABox deletions in the context of RDFS
  - [Calvanese et al., ISWC2010] ABox & TBox insertions in the context of DL-Lite (incl. inconsistency repair)

  Also related:
  - Deductive DBs: [Gupta et al., SIGMOD93]: **DRed (delete and re-derive)**, applied by [Kotowski et al.2011] and [Urbani et al. 2013] in the context of RDF/RDFS…
  - KB evolution, Belief revision, etc.: Various works in classical AI and philosophy

Particularly, none of these considers the **interplay** between **DELETE**, **INSERT** based on a joint **WHERE** clause as in SPARQL

Particularly:

- RDF Stream Reasoning/Stream Processing could be likewise viewed as **stream of updates** adding or removing triples, but:

- What does it meant to remove implicit triples?
- Is overdeletion of effects (Dred) always the right thing to do?

What about the **interplay** between **DELETE**, **INSERT** based on a joint **WHERE** clause as in SPARQL Updates?

Our initial thoughts on this problem…

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

ISWC 2014
Riva del Garda - Trentino, Italy

Updating RDFS ABoxes and TBoxes in SPARQL

Albin Ahmeti[1], Diego Calvanese[2], and Axel Polleres[3]

[1] Vienna University of Technology, Favoritenstraße 9, 1040 Vienna, Austria
[2] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
[3] Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Abstract. Updates in RDF stores have recently been standardised in the SPARQL 1.1 Update specification. However, computing entailed answers by ontologies is usually treated orthogonally to updates in triple stores. Even the W3C SPARQL 1.1 Update and SPARQL 1.1 Entailment Regimes specifications explicitly exclude a standard behaviour for entailment regimes other than simple entailment in the context of updates. In this paper, we take a first step to close this gap. We define a fragment of SPARQL basic graph patterns corresponding to (the RDFS fragment of) *DL-Lite* and the corresponding SPARQL update language, dealing with updates both of ABox and of TBox statements. We discuss possible semantics along with potential strategies for implementing them. In particular, we treat both, (i) materialised RDF stores, which store all entailed triples explicitly, and (ii) reduced RDF Stores, that is, redundancy-free RDF stores that do not store any RDF triples (corresponding to *DL-Lite* ABox statements) entailed by others already. We have implemented all semantics prototypically on top of an off-the-shelf triple store and present some indications on practical feasibility.

EFMD EQUIS ACCREDITED

Exploring possible ABox update semantics

- Materialized-preserving semantics
  - $\boldsymbol{Sem_0^{mat}}$ … baseline semantics
  - $\boldsymbol{Sem_{1a}^{mat}}$
  - $\boldsymbol{Sem_{1b}^{mat}}$

    inspired by DRed: *delete* (incl. **effects**) and re-derive new effects upon inserts
  - $\boldsymbol{Sem_2^{mat}}$ … delete incl. **causes** and rewrite upon inserts

Sem_0^{mat}: *Naïve Update followed by re-materialization*

G

TBOX

S	P	O
:Mother	rdfs:subClassOf	:Parent
:hasMother	rdfs:subPropertyOf	:hasParent
:hasMother	rdfs:range	:Mother
:hasParent	rdfs:domain	:Child
:hasParent	rdfs:range	:Parent
…		

```
DELETE { ?X a :Child . }
INSERT { ?Y a :Mother . }
WHERE { ?X :hasParent ?Y . }
```

```
DELETE { :marie a :Child . }
INSERT { :maria_t a :Mother . }
```

?X=:marie
?Y=:maria_t

ABOX-materialized

S	P	O
:marie	:hasMother	:maria_t
:marie	:hasParent	:maria_t
:maria_t	a	:Mother
:maria_t	a	:Parent

materialize(G)

S	P	O
:marie	:hasMother	:maria_t
:marie	a	:Child
:marie	:hasParent	:maria_t
:maria_t	a	:Mother
:maria_t	a	:Parent

No effect!

Alternative Materialized-pres. semantics

- ***Sem$_{1a}$^{mat}***
  - "Delete and rederive"

$$G^{Sem^{mat}_{1a}}_{u(P_d,P_i,P_w)} = materialize(\mathcal{T} \cup (\mathcal{A} \setminus materialize(\mathcal{T} \cup \mathcal{A}_d)) \cup \mathcal{A}_i)$$

$$\mathcal{A}_d = \bigcup_{\theta \in ans(P_w,G)} gr(P_d\theta)$$

$$\mathcal{A}_i = \bigcup_{\theta \in ans(P_w,G)} gr(P_i\theta)$$

1. DELETEs triples **incl. Effects**
2. INSERT triples
3. Re-materialize

Sem_{1a}^{mat}: *Delete and rederive*

G

TBOX

S	P	O
:Mother	rdfs:subClassOf	:Parent
:hasMother	rdfs:subPropertyOf	:hasParent
:hasMother	rdfs:range	:Mother
:hasParent	rdfs:domain	:Child
:hasParent	rdfs:range	:Parent
…		

DELETE { :marie :hasMother :maria_t. }

DELETE { :marie :hasMother :maria_t .
:marie :hasParent :maria_t .
:marie a Child .
:maria_t a Mother.
:maria_t a Parent. }

ABOX-materialized

S	P	O

materialize(G)

S	P	O

May be viewed quite "radical"

Sem_{1a}^{mat}: *Delete and rederive*

G

TBOX

S	P	O
:Mother	rdfs:subClassOf	:Parent
:hasMother	rdfs:subPropertyOf	:hasParent
:hasMother	rdfs:range	:Mother
:hasParent	rdfs:domain	:Child
:hasParent	rdfs:range	:Parent
…		

DELETE { :marie :hasParent :maria_t. }

DELETE { :marie :hasParent :maria_t .
:marie a Child .
:maria_t a Parent. }

ABOX-materialized

S	P	O
:marie	:hasMother	:maria_t
:maria_t	a	:Mother

materialize(G)

S	P	O
:marie	:hasMother	:maria_t
:marie	a	:Child
:marie	:hasParent	:maria_t
:maria_t	a	:Mother
:maria_t	a	:Parent

Again: no effect!

Alternative Materialized-pres. semantics

- ***Sem*$_{\mathbf{1b}}^{\boldsymbol{mat}}$**
  - Variant of Sem$_{1a}$, that makes a distinction between *explicit* and *implicit* triples.
  - Re-materialization from scratch from $\mathcal{A}'_{expl}$

$$G^{Sem^{mat}_{1b}}_{u(P_d,P_i,P_w)} = \mathcal{T} \cup \mathcal{A}'_{expl} \cup \mathcal{A}'_{impl}$$

$$\mathcal{A}'_{expl} = (\mathcal{A}_{expl} \setminus \mathcal{A}_d) \cup \mathcal{A}_i$$

$$\mathcal{A}'_{impl} = materialize(\mathcal{A}'_{expl} \cup \mathcal{T}) \setminus \mathcal{T}$$

Sem_{1b}^{mat}: *Delete and rerive with separating "explicit" and "implicit" ABox*

DELETE { :marie :hasParent :maria_t. }

G

TBOX

S	P	O
:Mother	rdfs:subClassOf	:Parent
:hasMother	rdfs:subPropertyOf	:hasParent
:hasMother	rdfs:range	:Mother
:hasParent	rdfs:domain	:Child
:hasParent	rdfs:range	:Parent
…		

$ABox_{expl}$

S	P	O
:marie	:hasMother	:maria_t

$ABox_{impl}$

S	P	O
:marie	:hasMother	:maria_t
:marie	a	:Child
:marie	:hasParent	:maria_t
:maria_t	a	:Mother
:maria_t	a	:Parent

materialize(G)

$Abox'_{impl}$

S	P	O
:marie	:hasMother	:maria_t
:marie	a	:Child
:marie	:hasParent	:maria_t
:maria_t	a	:Mother
:maria_t	a	:Parent

Again: no effect!

Alternative Materialized-pres. semantics

- ***Sem₂^mat***
  - **Delete** the instantiations of P_d **plus all their causes**;
  - **Insert** the instantiations of P_i **plus all their effects**.

$$G^{Sem_2^{mat}}_{u(P_d,P_i,P_w)} = G_{u(P_d^{caus},P_i^{eff},\{P_w\}\{P_d^{fvars}\})}$$

$$P_d^{fvars} = \{\texttt{?x a rdfs:Resource.} \mid \text{for each } \texttt{?x} \in Var(P_d^{caus_{P_d}}) \setminus Var(P_d)\}$$

3. SPARQL 1.1 Update & Entailment as a basis for Stream Reasoning Semantics? – Discussion:

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

- A first step to close the gap left by the current standards (**SPARQL1.1 Update** vs. **SPARQL1.1 Entailment Regimes**)

  - Seemingly no “one-size fits all” semantics: Particularly, DRed not intuitive in all cases!
  - Query rewriting may in some cases be more efficient then re-materialization
  - Non-intuitive corner cases in **each possible** semantics

    → depends on use case?

Which stream scenarios scenarios warrant which semantics?

EFMD EQUIS ACCREDITED