



WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS



SPARQL1.1 Updates and Entailment - Why the specification is silent about their interaction

Based on joint work with: Albin Ahmeti, Diego Calvanese, Vadim Savenkov

Axel Polleres

web: <http://polleres.net>

twitter: @AxelPolleres

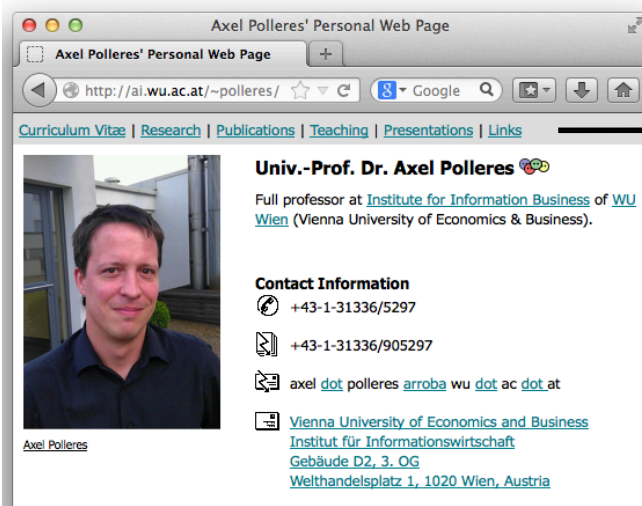
1. RDF & Linked Data – as a Data abstraction layer of Databases published on the Web
2. SPARQL - Data – as an abstraction layer of Data services on the Web
3. Two new features in SPARQL 1.1:
 - Entailment regimes
 - SPARQL Update
- 4. How do they interact?**
 - **Possible Semantics & Examples**
5. Discussion

From the HTML Web...

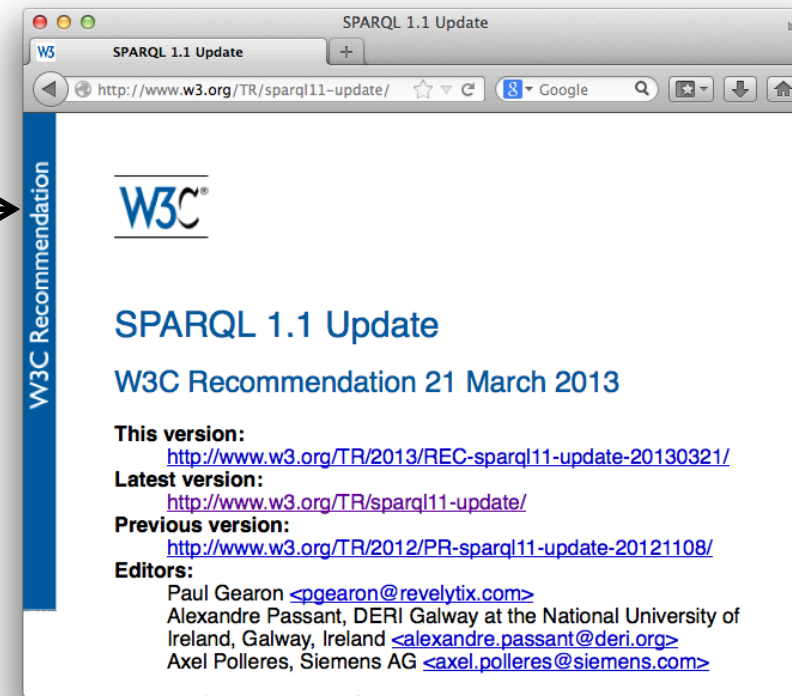
- Globally Unique identifiers
- Links between **Documents** (href)
- A common protocol

URIs

HTTP



href



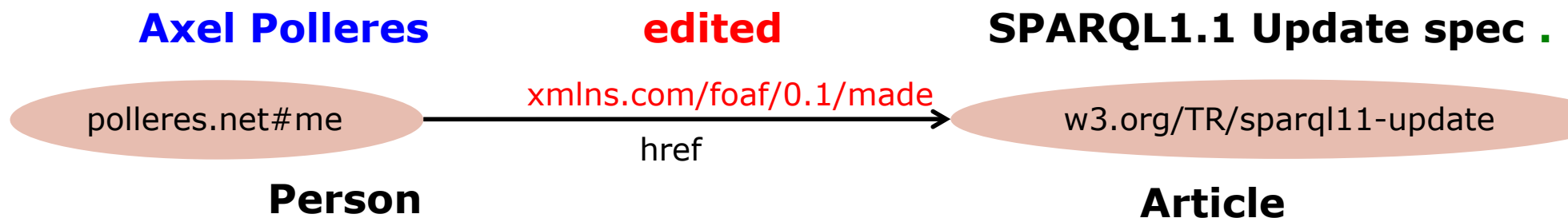
From the Web to Web (Linked) Data

- Globally Unique identifiers
- Typed Links (=relations) between Entities
- A common protocol

URIs

RDF

HTTP



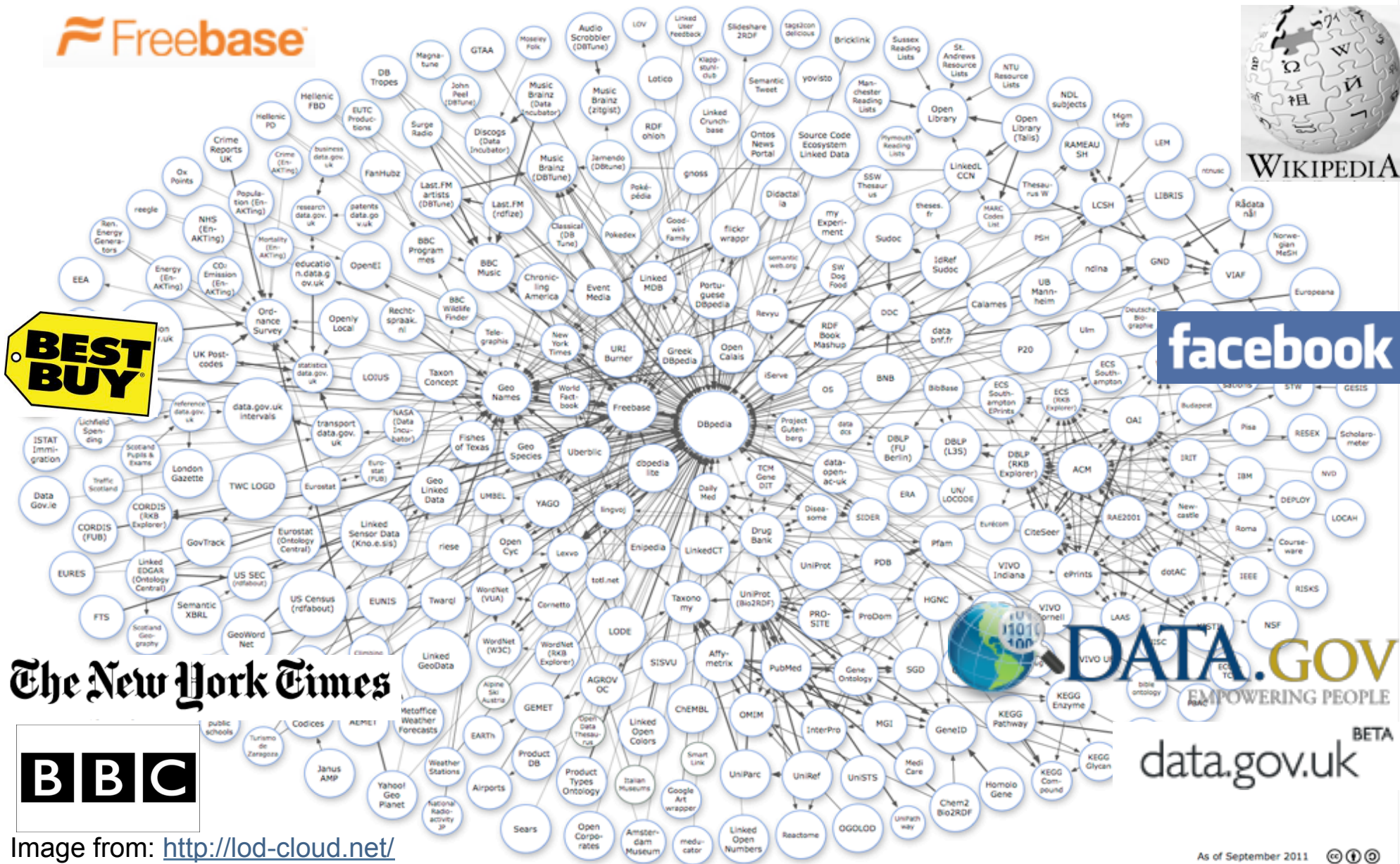
→ a universal graph-based data format...

(note that e.g. any relational data can be decomposed into such triples)

Any data can be represented and linked using RDF



WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF



The New York Times



Image from: <http://lod-cloud.net/>

data.gov.uk **BETA**

Linked Data has moved from academia
to industry over the last few years...

bing™

B B C

Google
UK

The New York Times

IBM

Watson

Y!
YAHOO!™

BEST
BUY®

Google Knowledge Graph

[News for venice italy](#)

[Tomado tears through parts of Venice, Italy \(VIDEOS\)](#)

[Washington Post \(blog\)](#) - 16 hours ago

A rare tornado (or waterspout, when over water) swept over several islands (Lido, Sant'Elena and Sant'Erasmus) off Venice's lagoon earlier ...

[Italy putting brakes on excitement](#)

[London Free Press](#) - 14 hours ago

[Italy could be hit by Spanish contagion](#)

[Economic Times](#) - 1 day ago

[Venice - Wikipedia, the free encyclopedia](#)

en.wikipedia.org/wiki/Venice

Venice (**Italian:** Venezia [veˈnɛttsja] (listen), **Venetian:** Venexia [veˈnɛsja]) is a city in northeast Italy sited on a group of 118 small islands separated by canals ...

↳ [History of the Republic of Venice - Venice, Los Angeles - Grand Canal](#)

[Venice Vacations, Tourism and Venice, Italy Travel Reviews ...](#)

www.tripadvisor.com/Tourism-g187870-Venice_Veneto-Vacations.h...

Venice Vacations: With 130000 reviews of **Venice, Italy** travel resources, TripAdvisor is the source for Venice information.

[ItalyGuides.it: Virtual tour of Venice, Italy - travel information and city ...](#)

www.italyguides.it/us/venice_italy/venice_travel.htm

Venice tourism and travel information: transport, attractions, maps, travel advice, pictures, audio guides, airport information, activities, hotels and more in **Venice**, ...

[Official website of the Municipality of Venice - Comune di Venezia](#)

www.comune.venezia.it/flex/cm/pages/ServeBLOB.php/L/EN/.../1

Official website of the Municipality of **Venice, Italy**. News, information and tools available to citizens and visitors.

Venice



Venice is a city in northeast Italy sited on a group of 118 small islands separated by canals and linked by bridges. It is located in the marshy Venetian Lagoon which stretches along the shoreline between the mouths of the Po and the Piave Rivers. [Wikipedia](#)

Area: 159 sq miles (412 km²)

Weather: 72° F, Wind E at 5 mph, 50% Humidity

Local time: 12:13pm Wednesday (CEST)

Points of interest



Grand Canal of Venice



Piazza San Marco



Saint Mark's Basilica



Rialto Bridge



Doge's Palace

[Report a problem](#)

The “Semantic Web” promise...



*“If **HTML** and the Web made all the online documents look like one huge **book**, **RDF**, **schema** and **inference** languages will make all the data in the world look like one huge **database**”*

Tim Berners-Lee, 1999

Try the following in google:

- *“Children of Austrian rulers married to French rulers...?”*



The necessary information is available in RDF – Dbpedia: Wikipedia as a Data Service...

W Marie Antoinette - Wikipedi...
 http://en.wikipedia.org/wiki/Marie_Antoinette

WIKIPEDIA
 The Free Encyclopedia

Marie Antoinette

From Wikipedia, the free encyclopedia

For other uses, see *Marie Antoinette (disambiguation)*.

Marie Antoinette (/mariˈæntwəˈneɪ/ or /æntwɑːneɪ/; French: [maʁi ɑ̃twanɛt]; baptised **María Antonia Josepha Johanna**;^[1] 2 November 1755 – 16 October 1793), born an **Archduchess of Austria**, was **Dauphine of France** from 1770 to 1774 and **Queen of France and Navarre** from 1774 to 1792. She was the fifteenth and penultimate child of **Holy Roman Empress Maria Theresa** and **Emperor Francis I**.

In April 1770, upon her marriage to **Louis-Auguste, Dauphin of France**, she became Dauphine of France. She assumed the title Queen of France and of Navarre when her husband ascended the throne as Louis XVI upon the death of his grandfather **Louis XV** in May 1774. After seven years of marriage, she gave birth to a daughter, **Marie-Thérèse Charlotte**, the first of her four children.

Initially charmed by her personality and beauty, the French people eventually came to dislike her, accusing "L'Autrichienne" (which literally means *the Austrian (woman)*, but also suggests the French word "chienne", meaning *bitch*) of being profligate, promiscuous,^[2] and of harbouring sympathies for France's enemies, particularly Austria, her country of origin.^[3] The **Diamond Necklace incident** damaged her reputation further. She later became known as *Madame Déficit* because France's financial crisis was blamed on her lavish spending.

Marie Antoinette of Austria

María Antonia Josepha Joanna, Queen of France and Navarre, by *Louise Élisabeth Vigée Le Brun*, ca. 1779.

About: Marie Antoinette
 http://dbpedia.org/page/Marie_Antoinette

About: **Marie Antoinette**

S	P	O
:marie	:hasMother	:maria_t
:marie	:hasSpouse	:louis
:maria_t	a	:RulerOfAustria
:louis	a	:DauphinOfFrance

personality and beauty, the French people eventually came to dislike her, accusing "L'Autrichienne" (the Austrian (woman), but also suggests the French word "chienne", meaning bitch) of being profligate, promiscuous, and of harbouring sympathies for France's enemies, particularly Austria, her country of origin. The Diamond Necklace incident damaged her reputation further, although she was completely innocent in this affair. She later became known as Madame Déficit because France's financial crisis was blamed on her lavish spending. The royal family's flight to the United States during the French Revolution had disastrous effects on French popular opinion: Louis XVI was deposed and the monarchy abolished in 1792; the royal family was subsequently imprisoned at the Temple Prison. Eight months after her husband's death, Marie Antoinette was herself tried, convicted by the Revolutionary Tribunal of treason to the prince and executed by guillotine on 16 October 1793. Long after her death, Marie Antoinette is often cited in popular culture and a major historical figure, being the subject of several books, films and other media. Academics and scholars have deemed her frivolous and superficial, and have attributed the start of the French Revolution to her; however, others have claimed that she was treated unjustly and that views of her should be re-evaluated.

dbpedia-owl:activeYearsEndYear • 1792-01-01 (xsd:date)
 dbpedia-owl:activeYearsStartYear • 1774-01-01 (xsd:date)
 dbpedia-owl:alias • Maria Antonia Josefa Johanna von Habsburg-Lothringen
 dbpedia-owl:birthDate • 1755-11-02 (xsd:date)
 dbpedia-owl:birthPlace • dbpedia:Vienna

SPARQL: query language for RDF

- SPARQL offers a standard protocol/service interface to data offering services like DBpedia!

```
SELECT ?X ?P ?S WHERE {  
  ?X :hasMother ?P . ?X :hasSpouse ?S .  
  ?P a RulerOfAustria . ?S a :DauphinsOfFrance .  
}
```



X	P	S
http://dbpedia.org/resource/Marie_Antoinette	http://dbpedia.org/resource/Maria_Theresa	http://dbpedia.org/resource/Louis_XVI_of_France
http://dbpedia.org/resource/Elisabeth_of_Austria,_Queen_of_France	http://dbpedia.org/resource/Maximilian_II,_Holy_Roman_Emperor	http://dbpedia.org/resource/Charles_IX_of_France
http://dbpedia.org/resource/Marie_Antoinette	http://dbpedia.org/resource/Francis_I,_Holy_Roman_Emperor	http://dbpedia.org/resource/Louis_XVI_of_France

SPARQL 1.1

- SPARQL 1.0 (2008): SQL "look-and-feel" for RDF and Linked data
- SPARQL 1.1 (2013): adding demanded features
- Added two new features:
 - Use of implicit information: SPARQL 1.1 Entailment regimes
 - Provide an update interface: SPARQL 1.1 Update

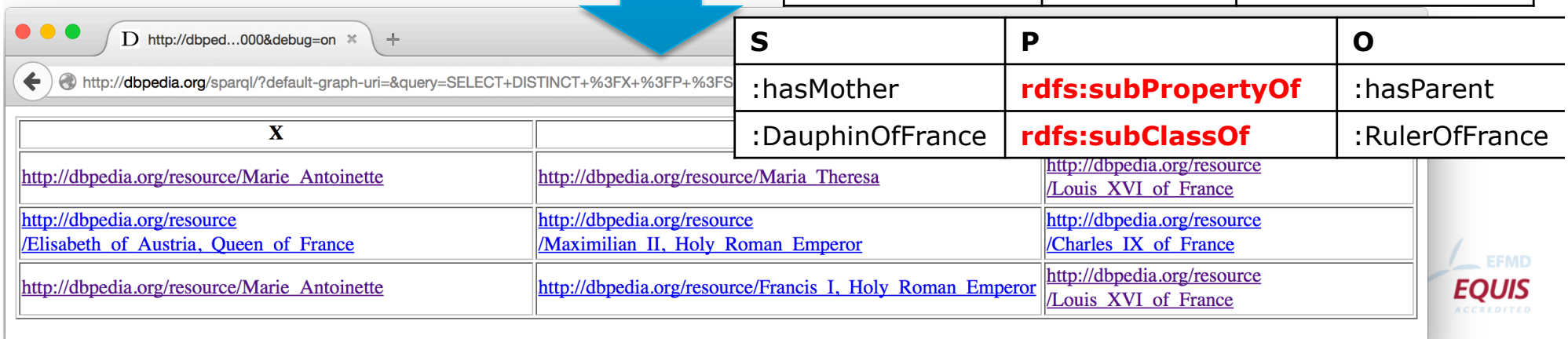
SPARQL1.1 Entailment Regimes:

- Make use of ontological meta-information (RDFS and OWL):

```

SELECT ?X ?P ?S WHERE {
  ?X :hasParent ?P . ?X :hasSpouse ?S .
  ?P a RulerOfAustria . ?S a :RulerOfFrance .
}
    
```

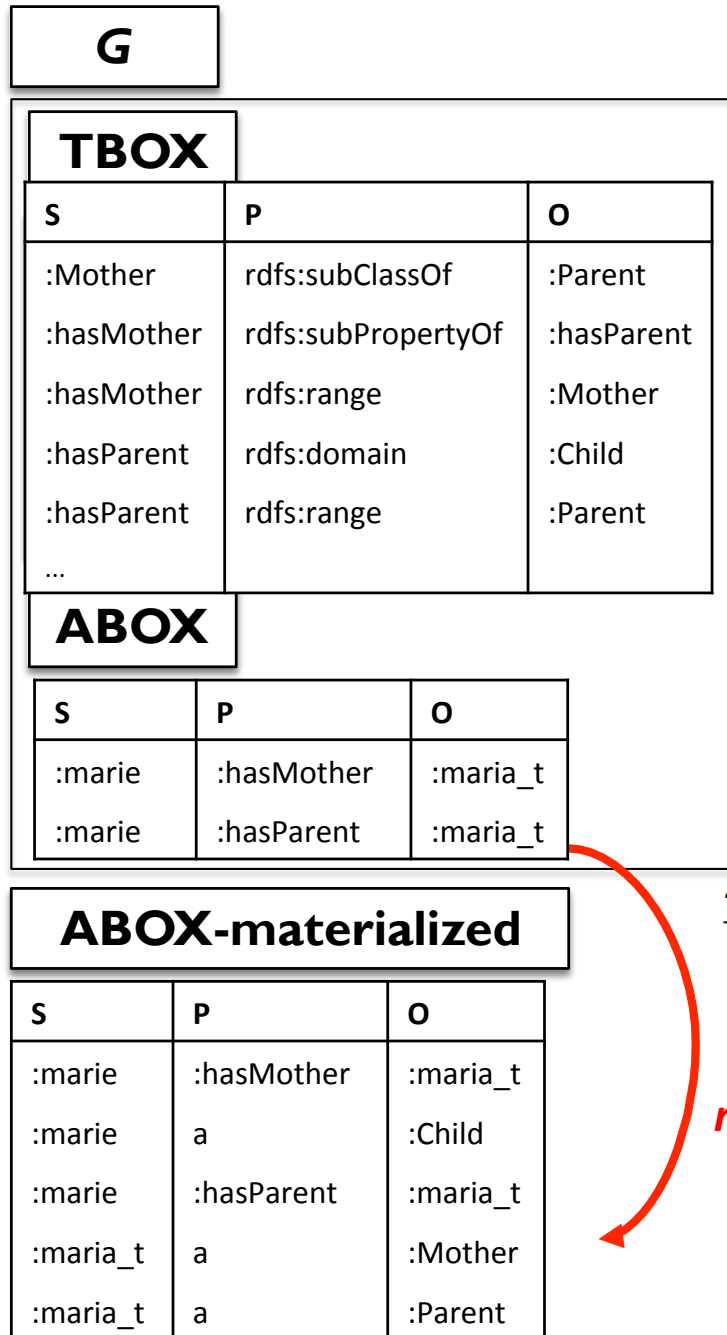
S	P	O
:marie	:hasMother	:maria_t
:marie	:hasSpouse	:louis
:maria_t	rdf:type	:RulerOfAustria
:louis	rdf:type	:DauphinOfFrance



X	S	P	O
http://dbpedia.org/resource/Marie_Antoinette	http://dbpedia.org/resource/Maria_Theresa	http://dbpedia.org/resource/Louis_XVI_of_France	
http://dbpedia.org/resource/Elisabeth_of_Austria,_Queen_of_France	http://dbpedia.org/resource/Maximilian_II,_Holy_Roman_Emperor	http://dbpedia.org/resource/Charles_IX_of_France	
http://dbpedia.org/resource/Marie_Antoinette	http://dbpedia.org/resource/Francis_I,_Holy_Roman_Emperor	http://dbpedia.org/resource/Louis_XVI_of_France	

How are Entailment Regimes typically implemented in SPARQL engines?

SPARQL Query answering under (RDFS) Entailment:



[Munoz et al., ESWC2007] *Minimal RDFS (ABOX-) Inference Rules*

$$\frac{?C \text{ rdfs:subClassOf } ?D. \quad ?S \text{ a } ?C.}{?S \text{ a } ?D.}$$

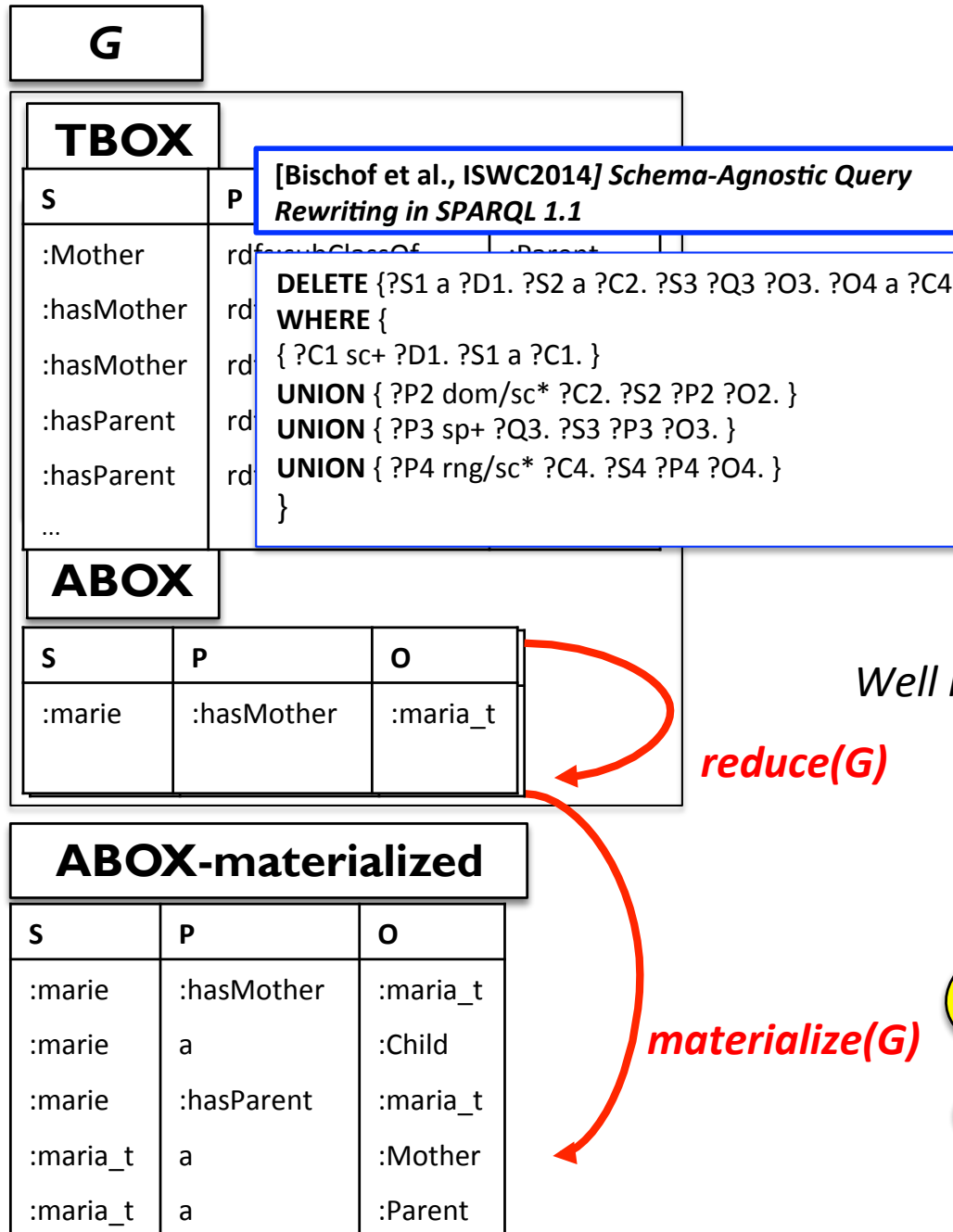
$$\frac{?P \text{ rdfs:domain } ?C. \quad ?S ?P ?O.}{?S \text{ a } ?C.}$$

$$\frac{?P \text{ rdfs:subPropertyOf } ?Q. \quad ?S ?P ?O.}{?S ?Q ?O.}$$

$$\frac{?P \text{ rdfs:range } ?C. \quad ?S ?P ?O.}{?O \text{ a } ?C.}$$

materialize(G)

SPARQL Query answering under RDFS Entailment:



SELECT ?Y
WHERE {
 { :marie :hasParent ?Y . }
 UNION
 { :marie :hasMother ?Y . }
 UNION
 { :marie :hasFather ?Y . }
}

Well known:

$$ans(rewrite(q, T), reduce(G)) = ans(q, materialize(reduce(G)))$$

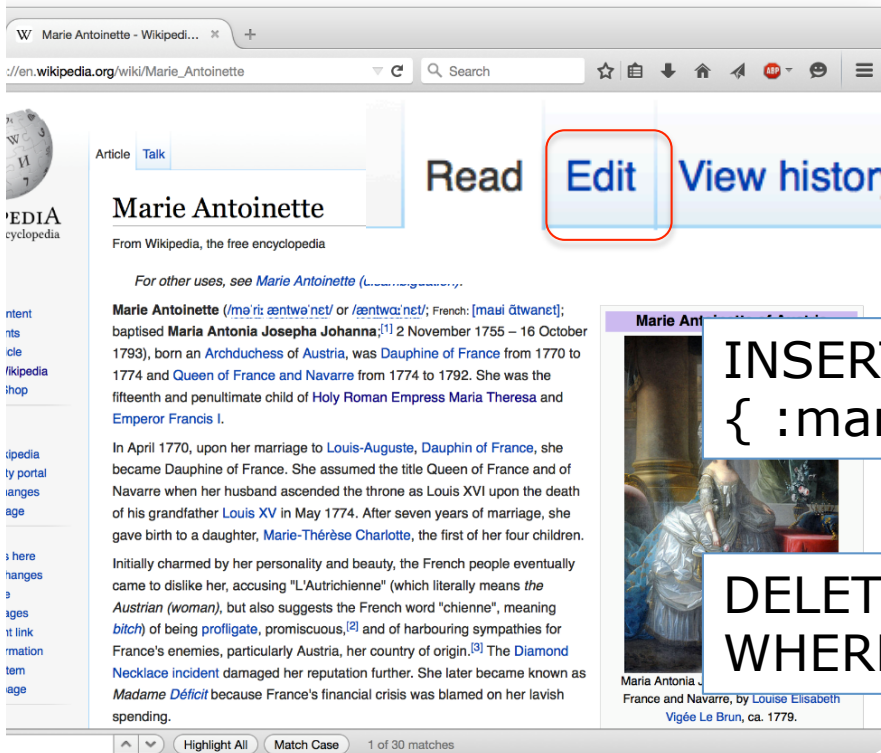
SELECT ?Y
WHERE { :marie :hasParent ?Y . }

SPARQL 1.1

- SPARQL 1.0 (2008): SQL "look-and-feel" for RDF and Linked data
- SPARQL 1.1 (2013):
- Added two new features:
 - Use of implicit information: SPARQL 1.1 Entailment regimes
 - Provide an update interface: SPARQL 1.1 Update

Many off-the-shelf engines support at least RDFS in one or the other way

Data Services on the Web allow updates:



Examples:

INSERT
 { :marie :hasChild :marie_ +
 _France . }

DELETE { ?X ?P ~
 WHERE { ?X :maria_t }

DELETE { ?X :hasParent :maria_t . }
 INSERT { ?X :hasMother :maria_t . }
 WHERE { ?X :hasParent :maria_t . }

What do these updates mean in a materialized or reduced store?

Status:

- Standardization of **SPARQL 1.1 Update**, and **SPARQL 1.1 Entailment Regimes** with triple stores implementing those standards (**rewriting-** or **materialization-based**)
- Emerging need for **a more systematic approach** of dealing with SPARQL 1.1 Update over ABoxes & Tboxes

This Talk: We have run into different directions, but we haven't reached the goal yet!



Nothing endures but change. ¹⁸ Heraclitus

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

Our initial thoughts on this problem...



Updating RDFS ABoxes and TBoxes in SPARQL

Albin Ahmeti¹, Diego Calvanese², and Axel Polleres³

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.

Let's start with a minimizing expectations:

- Discuss possible update semantics in the context of **materialized** and **reduced** stores & **RDFS**:
 - Only ABox updates, TBox fixed
 - Use "minimal" RDFS as TBox language (without axiomatic triples, blank nodes) ... i.e., DL-Lite_{RDFS}
 - Restrict on BGPs to **only** allow **ABox** Insert/Deletes

```
INSERT { :marie :hasMother ?Y }  
WHERE  
{ :marie :hasParent ?Y }
```

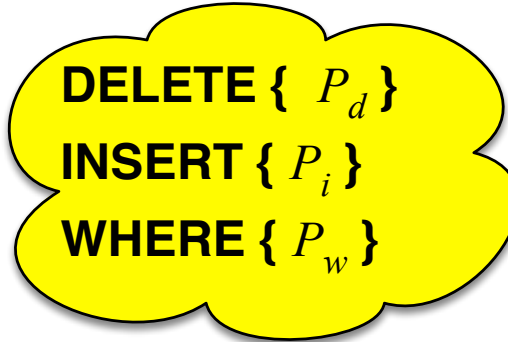
```
INSERT { :marie ?Y :foo }  
WHERE  
{ :marie rdf:type ?Y }
```

- *Even in this restricted setting (RDFS) defining a reasonable update semantics under entailments turns out to be challenging!*

Exploring possible ABox update semantics for SPARQL+RDFS

- Materialized-preserving semantics
 - Sem_0^{mat} ... baseline semantics
 - Sem_{1a}^{mat} } inspired by DRed: *delete* (incl. **effects**) and re-derive new effects upon inserts
 - Sem_{1b}^{mat} }
 - Sem_2^{mat} ... delete incl. **causes** and rewrite upon inserts
- Reduced-preserving semantics
 - Sem_0^{red} ... baseline semantics
 - Sem_1^{red} ... delete incl. **causes** followed by reduce

Baseline semantics for materialized stores:



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

$$G_{u(P_d, P_i, P_w)}^{Sem_0^{mat}} = \text{materialize}(G_{u(P_d, P_i, P_w)})$$

Sem₀^{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}
 - “(Over-)delete and rederive”

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

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

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

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

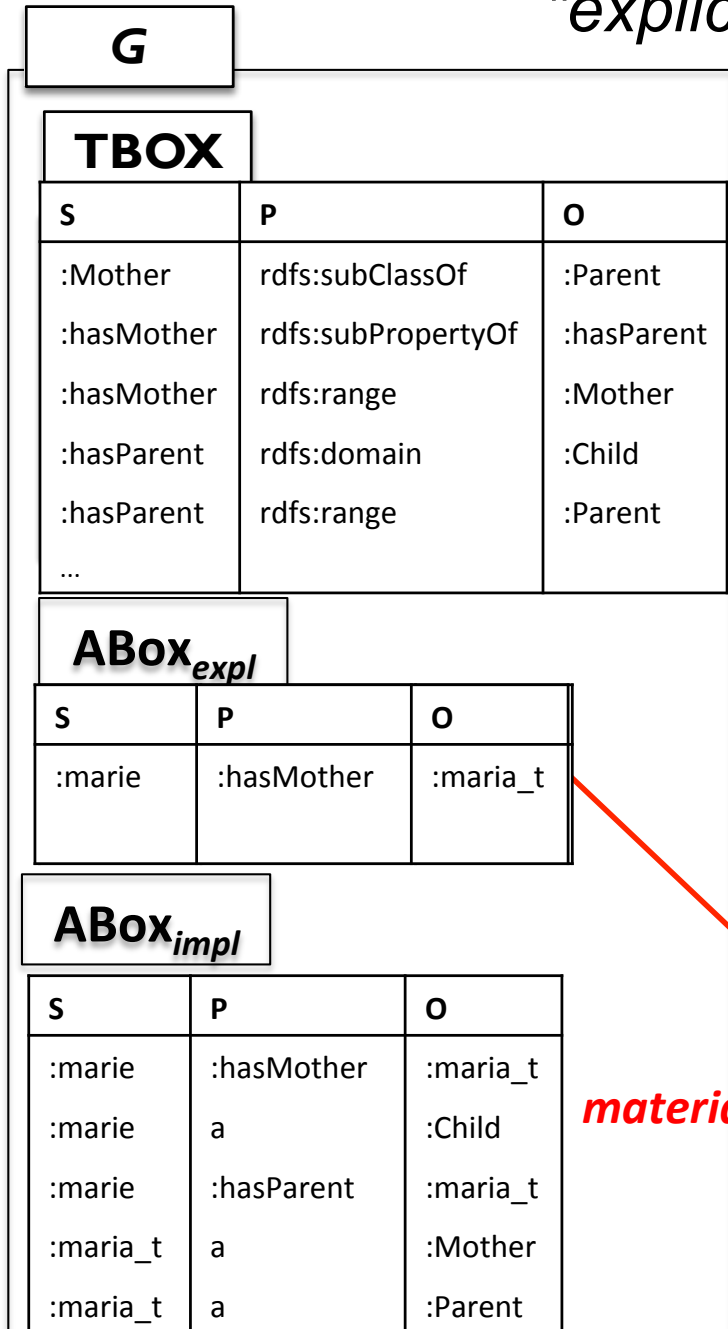
- **Sem_{1b}^{mat}**
 - Variant of Sem_{1a} , that makes a distinction between *explicit* and *implicit* triples.
 - Re-materialization from scratch from \mathcal{A}'_{expl}

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

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

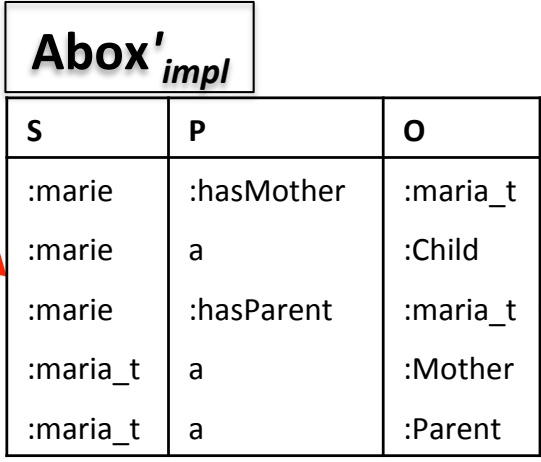
$$\mathcal{A}'_{impl} = \text{materialize}(\mathcal{A}'_{expl} \cup \mathcal{T}) \setminus \mathcal{T}$$

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



DELETE { :marie :hasParent :maria_t. }

materialize(G)



Again: no effect!

Alternative Materialized-pres. semantics

- Sem_2^{mat}
 - Delete the instantiations of P_d **plus all their causes**;
 - Insert the instantiations of P_i **plus all their effects**.

$$G_u^{Sem_2^{mat}}(P_d, P_i, P_w) = G_u(P_d^{caus}, P_i^{eff}, \{P_w\} \{P_d^{fvars}\})$$

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

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

S	P	O
:marie	:hasMother	:maria_t
:marie	:hasParent	:maria_t

ABOX-materialized

S	P	O
:maria_t	a	:Mother
:maria_t	a	:Parent

materialize(G)

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

rewrite(u,T)

DELETE { ?X a :Child. ?X :hasFather ?x1.
 ?X :hasMother ?x2. ?X :hasParent ?x3. }
 INSERT { ?Y a :Mother. ?Y a :Parent. }
 WHERE { { ?X :hasMother ?Y. }
 { ?x1 a rdfs:Resource. ?x2 a rdfs:Resource.
 ?x3 a rdfs:Resource. } }

DELETE { :marie a :Child.
 :marie :hasFather :maria_t, :marie, :louis
 :marie :hasMother :maria_t, :marie, :louis
 :marie :hasParent :maria_t, :marie, :louis ... }
 INSERT { :maria_t a :Mother . :maria_t a Parent . }

?X=:marie ?Y=:maria_t
 ?x1=?x2=?x3=*

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

S:ja	P	O
:marie	a	:Child
:maria_t	a	:Mother
:maria_t	a	:Parent
:marie	:hasParent	:maria_t
:louis	a	:Father
:marie	:hasParent	:louis

```
DELETE {}
INSERT {:marie :hasMother :maria_t;:hasFather :louis}
WHERE {};
```

```
DELETE {:marie :hasMother :maria_t;:hasFather :louis }
INSERT {}
WHERE {};
```

rewrite(u1,T)

rewrite(u2,T)

```
DELETE {}
INSERT {:marie :hasMother :maria_t;:hasFather :louis,
:hasParent :maria_t, :hasParent :louis.
:marie a :Child . :maria_t a :Mother, :Parent.
:louis a :Father, Parent.} WHERE {};
```

```
DELETE {:marie :hasMother :maria_t;:hasFather :louis }
INSERT {}
WHERE {};
```

DELETE following INSERT is NOT idempotent!
 → Leaves "Dangling effects"

Baseline semantics

- Sem_0^{red}
 - Naïve Update followed by re-reduce

$$G_{u(P_d, P_i, P_w)}^{Sem_0^{red}} = red(G_{u(P_d, P_i, P_w)})$$

Sem_0^{red} : Naïve Update followed by re-reduce

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 :hasMother ?Y . }

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

?X=:marie
 ?Y=:maria_t

ABOX-reduced

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

reduce(G)

S	P	O
:marie	:hasMother	:maria_t

No effect!

Alternative Reduced-pres. semantics

- Sem_1^{red}
 - Delete the instantiations of P_d **plus all their causes**;
 - Insert the instantiations of P_i .
 - Followed by **re-reduce**

$$G_{u(P_d, P_i, P_w)}^{Sem_1^{red}} = red(G_{u(P_d^{caus}, P_i, \{rewrite(P_w)\} \{P_d^{fvars}\})})$$

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

Sem₁^{red}: Re-written Update followed by re-reduce

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-reduced

S	P	O
:maria_t	a	:Mother

S	P	O
:maria_t	a	:Mother

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

rewrite(u,T)

DELETE { ?X a :Child. ?X :hasFather ?x1.
 ?X :hasMother ?x2. ?X :hasParent ?x3. }
 INSERT { ?Y a :Mother. }
 WHERE { { ?X :hasMother ?Y. }
 { ?x1 a rdfs:Resource. ?x2 a rdfs:Resource.
 ?x3 a rdfs:Resource. } }

DELETE {joe a :Child.
 :marie :hasMother :maria_t, :louis,
 :marie :hasParent :maria_t, :louis,
 ... }
 INSERT { :maria_t a :Mother. }

?X=:marie ?
 Y=:maria_t
 ?x1=?x2=?x3=*

reduce(G)



Extensions

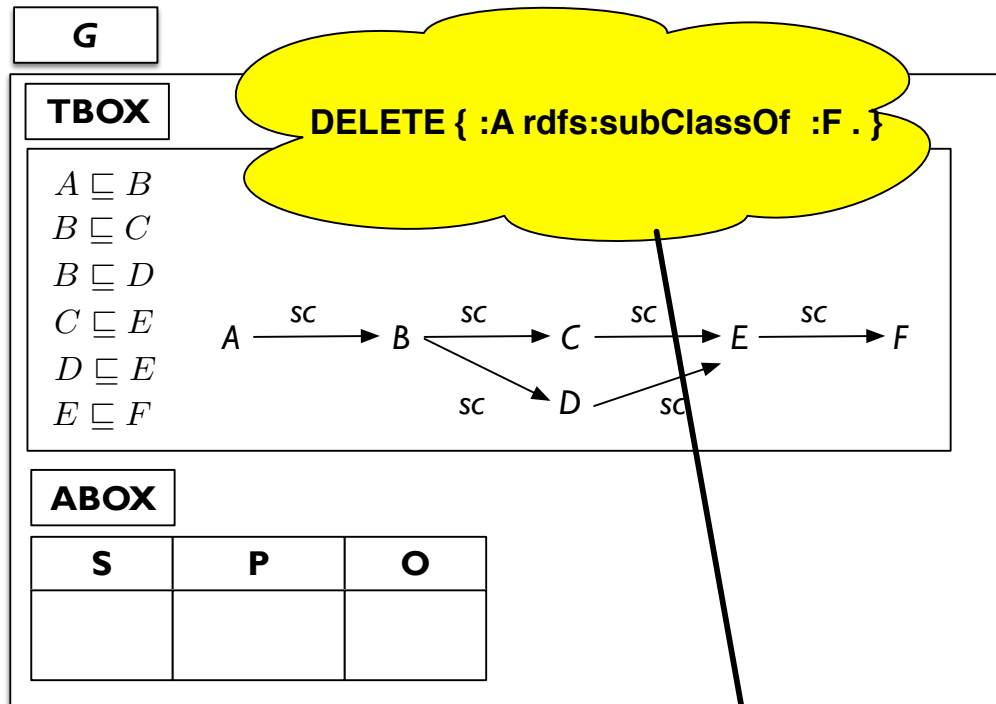
- TBox updates
- (ABox) Updates in the presence of ontologies beyond RDFS: dealing with inconsistencies.
- Experiments & Use cases
 - Bring mappings into play: Ontology-based data management

Extension 1: TBox updates

- In this setting
 - We expand BGPs to take into account TBox Inserts/Deletes – **general BGPs**
 - Tbox **inserts** are trivial in this setting of RDFS.
 - TBox **deletions** for RDFS are ambiguous
(need *minimal cuts*) [Gutierrez et al., ESWC2011]
- Opens various degrees of freedom... What if we consider a materialized (acyclic) Tbox?
 - We also use two RDFS rules **for TBox materialization.**

DELETE { :A rdfs:subClassOf ?C . }

$$\frac{?C \text{ sc } ?D. \quad ?D \text{ sc } ?E.}{?C \text{ sc } ?E.}$$
$$\frac{?P \text{ sp } ?Q. \quad ?Q \text{ sp } ?R.}{?P \text{ sp } ?R.}$$



materialize(G)

G-materialized

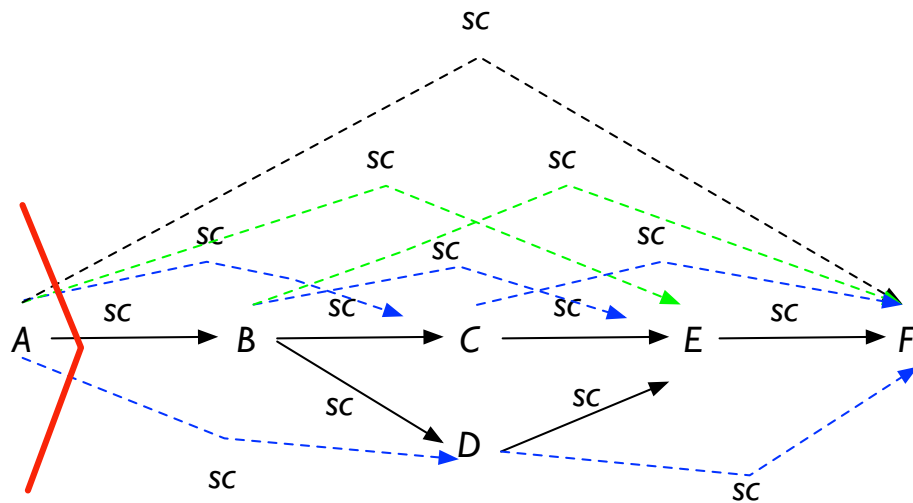
S	P	O
:A	rdfs:subClassOf	:B
:B	rdfs:subClassOf	:C
:B	rdfs:subClassOf	:D
:C	rdfs:subClassOf	:E
:E	rdfs:subClassOf	:F
:D	rdfs:subClassOf	:E
:A	rdfs:subClassOf	:C
:A	rdfs:subClassOf	:D
:A	rdfs:subClassOf	:E
:A	rdfs:subClassOf	:F
:B	rdfs:subClassOf	:E
:B	rdfs:subClassOf	:F
:C	rdfs:subClassOf	:F
:D	rdfs:subClassOf	:F

- ?
- DELETE { :A rdfs:subClassOf :B }
 - DELETE { :B rdfs:subClassOf :C.
:B rdfs:subClassOf :D . }
 - DELETE { :B rdfs:subClassOf :C.
:D rdfs:subClassOf :E . }
 - DELETE { :E rdfs:subClassOf :F }
 - ...

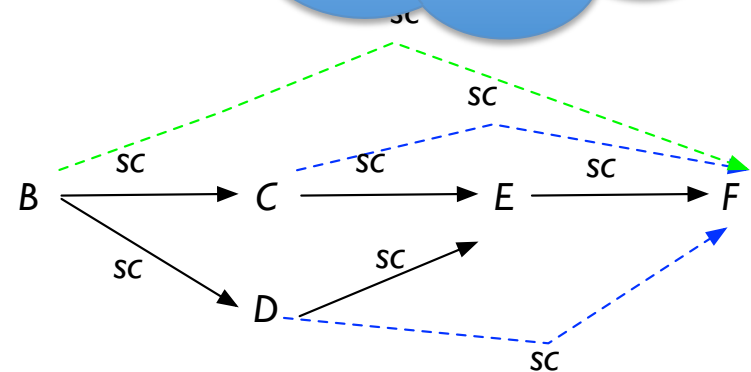
Minimal cuts
are
ambiguous!

Proposed TBox update semantics

- Outbound cut: Intuition



A

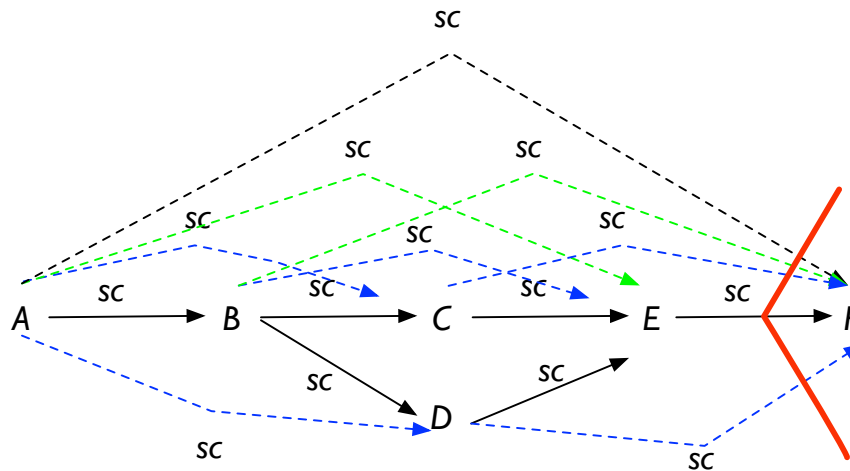


DELETE { :A rdfs:subClassOf ?X. }
 WHERE
 { :A rdfs:subClassOf ?X .
 ?X rdfs:subClassOf* :F. }

Idea: Can be implemented with SPARQL 1.1 property paths

Analogous alternative: *Sem*_{incut}

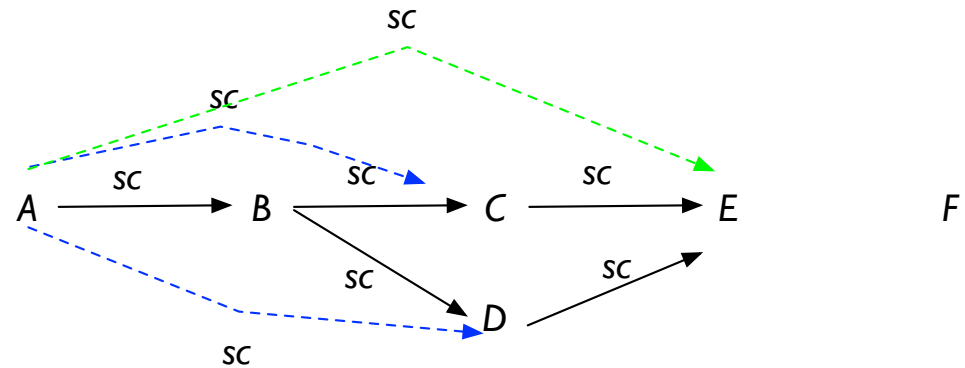
- Inbound cut: Intuition



DELETE { **?X** rdfs:subClassOf :F. }
WHERE
{ :A rdfs:subClassOf* **?X** .
 ?X rdfs:subClassOf :F. }

Downside of incut/outcut:

- **Only works for acyclic Tboxes**
- **Minimality of cut only guaranteed for single atomic updates**



Extension 2: Beyond RDFS



DL 2015

Dealing with Inconsistencies due to Class Disjointness in SPARQL Update

Albin Ahmeti^{1,3}, Diego Calvanese², Axel Polleres³, and Vadim Savenkov³

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. The problem of updating ontologies has received increased attention in recent years. In the approaches proposed so far, either the update language is restricted to (sets of) atomic updates, or, where the full SPARQL Update language is allowed, the TBox language is restricted to RDFS where no inconsistencies can arise. In this paper we discuss directions to overcome these limitations. Starting from a DL-Lite fragment covering RDFS and concept/class disjointness axioms, we define two semantics for SPARQL Update: under cautious semantics, inconsistencies are resolved by rejecting *all* updates potentially introducing conflicts; under brave semantics, instead, conflicts are overridden in favor of new information where possible. The latter approach builds upon existing work on the evolution of DL-Lite knowledge bases, setting it in the context of generic SPARQL updates.

Minimal ontology language extension: RDFS + class disjointness.

SPARQL updates: deal with inconsistency within *new knowledge*

:Monarch owl:disjointWith :Regent
:hasRegent rdfs:domain :Monarch
:hasRegent rdfs:range :Regent

$Monarch \sqsubseteq \neg Regent$
 $\exists hasRegent \sqsubseteq Monarch$
 $\exists hasRegent^- \sqsubseteq Regent$

"Unsafe" update → intrinsically inconsistent:

INSERT {?X :hasRegent ?Y} WHERE {?Y :signsDecreeInNameOf ?X}

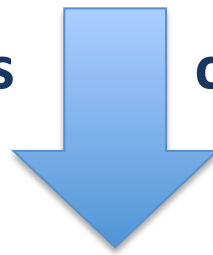
SPARQL updates: deal with inconsistency within *new knowledge*

:Monarch owl:disjointWith :Regent
:hasRegent rdfs:domain :Monarch
:hasRegent rdfs:range :NonRoyalRuler

"Unsafe" update:

```
INSERT {?X :hasRegent ?Y} WHERE {?Y :signsDecreeInNameOf ?X}
```

intrinsically Inconsistencies



can be caught by "safe rewriting"

```
INSERT{?X :hasRegent ?Y}  
WHERE{?Y :signsDecreeInNameOf ?X  
MINUS{ {?X1 :signsDecreeInNameOf ?Y}  
UNION {?X :signsDecreeInNameOf ?Y2}}}
```

Copies of the WHERE clause, variables renamed appropriately.

Rewritings: MINUS vs. FILTER NOT EXISTS

Safe rewriting via FILTER NOT EXISTS doesn't work:

```
DELETE{?Z a :Regent} INSERT{?X :hasRegent ?Y}
WHERE{ {?Y :signsDecreeInNameOf ?X}
        UNION {?V :signsDecreeInNameOf ?Z}}
```

Rewritings: MINUS vs. FILTER NOT EXISTS

Safe rewriting via FILTER NOT EXISTS doesn't work:

```
DELETE{?Z a :Regent} INSERT{?X :hasRegent ?Y}
WHERE{ {?Y :signsDecreeInNameOf ?X}
       UNION {?V :signsDecreeInNameOf ?Z}}
```

disjoint sets of variables

FILTER NOT EXISTS{

```
{?X1 :signsDecreeInNameOf ?Y} UNION {?V1 :signsDecreeInNameOf ?Z1}
}}
```

Exists whenever the WHERE clause is satisfied!

FILTER NOT EXISTS{

```
{?X :signsDecreeInNameOf ?Y2} UNION {?V2 :signsDecreeInNameOf ?Z2}
}}
```

Simply renaming the whole WHERE clause is not possible.

Rewritings: MINUS vs. FILTER NOT EXISTS

Safe rewriting via MINUS: works!

```
DELETE{?Z a :Regent} INSERT{?X :hasRegent ?Y}
WHERE{ {?Y :signsDecreeInNameOf ?X}
        UNION {?V :signsDecreeInNameOf ?Z}}
```

MINUS{

```
{?X1 :signsDecreeInNameOf ?Y} UNION {?V1 :signsDecreeInNameOf ?Z1}
}}
```

Extra union branches do not matter!

MINUS{

```
{?X :signsDecreeInNameOf ?Y2} UNION {?V2 :signsDecreeInNameOf ?Z2}
}}
```

- MINUS removes variable bindings of the WHERE clause *that can be combined with some result of the query in its right-hand side.*
- Only variables from the left-hand side of MINUS are "visible" in its right-hand side: great for our case!

SPARQL updates: deal with inconsistency w.r.t. the old knowledge

Adapt the Sem_2^{mat} (rewriting-based) semantics

- **Brave**: when in conflict, prefer new knowledge (cf. FastEvol [Calvanese et al 2010])
- **Cautious**: when in conflict, stick to the old knowledge
 - In batch updates, allow variable bindings only where the insert clause does not clash.
- **Fainthearted**: relaxation of cautious semantics
 - The same batch update might resolve clashes by deleting conflicting parts of the old knowledge!

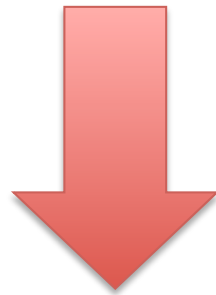
Example: Brave Sem_2^{mat}

:Monarch owl:disjointWith :Regent

:hasRegent rdfs:domain :Monarch

:hasRegent rdfs:range :NonRoyalRuler

INSERT{?X :hasRegent ?Y} WHERE{?Y :signsDecreeInNameOf ?X}



*Preprocess:
safe rewriting*

INSERT{?X :hasRegent ?Y}

WHERE{?Y :signsDecreeInNameOf ?X

MINUS { ?Y1 :signsDecreeInNameOf ?X }

UNION { ?Y :signsDecreeInNameOf ?X2 } }

Example: Brave Sem_2^{mat}

:Monarch owl:disjointWith :Regent
:hasRegent rdfs:domain :Monarch
:hasRegent rdfs:range :NonRoyalRuler

INSERT{?X :hasRegent ?Y} WHERE{?Y :signsDecreeInNameOf ?X}



**DELETE {?X a :Regent . ?X3 :hasRegent ?X .
?Y a :Monarch . ?Y :hasRegent ?Y3 }**

Potential
clashes

INSERT{?X :hasRegent ?Y . ?X a :Monarch . ?Y a :Regent}
WHERE{?Y :signsDecreeInNameOf ?X

MINUS{ ?Y1 :signsDecreeInNameOf ?X}

UNION {?Y :signsDecreeInNameOf ?X2}}

OPTIONAL{?X3 :hasRegent ?X}

OPTIONAL {?Y :hasRegent ?Y3}}

Bind variables
in DELETE

Example: Cautious Sem_2^{mat}

:Monarch owl:disjointWith :Regent

:hasRegent rdfs:domain :Monarch

:hasRegent rdfs:range :NonRoyalRuler

INSERT{?X :hasRegent ?Y} WHERE{?Y :signsDecreeInNameOf ?X}



Cautious Sem_2^{mat}

INSERT{?X :hasRegent ?Y . ?X a :Monarch . ?Y a :Regent}

WHERE{?Y :signsDecreeInNameOf ?X

MINUS{ {?Y1 :signsDecreeInNameOf ?X}

UNION {?Y :signsDecreeInNameOf ?X2}}

MINUS {{?X a :Regent} UNION {?Y a :Monarch}}

Do inserts only with non-clashing variable bindings
(assuming *materialized* store!)



Example: Fainthearted Sem_2^{mat}

Removes
some
clashes!

:Monarch owl:disjointWith :Regent
:hasRegent rdfs:domain :Monarch
:hasRegent rdfs:range :NonRoyalRuler

DELETE{?X :Regent}

INSERT{?X :hasRegent ?Y} WHERE{?Y :signsDecreeInNameOf ?X}



Fainthearted Sem_2^{mat}

DELETE{?X :Regent}

INSERT{?X :hasRegent ?Y . }

WHERE{?X :signsDecreeInNameOf ?Y

Handled by
delete

MINUS{ {?X1 :signsDecreeInNameOf ?X}
UNION {?Y :signsDecreeInNameOf ?Y2}}

~~MINUS { {?X a :Regent} UNION {?Y a :Monarch} }~~



Fainthearted semantics: pitfalls, e.g. clashes removed by **different** bindings

```
DELETE {?Z a :Regent}  
INSERT {?X :hasRegent ?Y}  
WHERE { ... }
```

$\mu_2 = [..., ?Z \mapsto :Arthur, ...]$: **Clears the clash!**

$\mu_1 = [?X \mapsto :Arthur, ...]$: **clash**

:Arthur a :Regent

- Atomic updates: for each variable binding μ of the WHERE clause either **both** delete **and** insert or **none**.
- Insert with μ_1 depends on the deletion with μ_2 .
- By atomicity, μ_1 also causes insertion (which might depend on the deletion with some μ_3 etc).

Idea: give up update atomicity. Delete for all μ_i of the WHERE pattern, insert only where not clashing; for this we have to "separate" DELETE and INSERT...



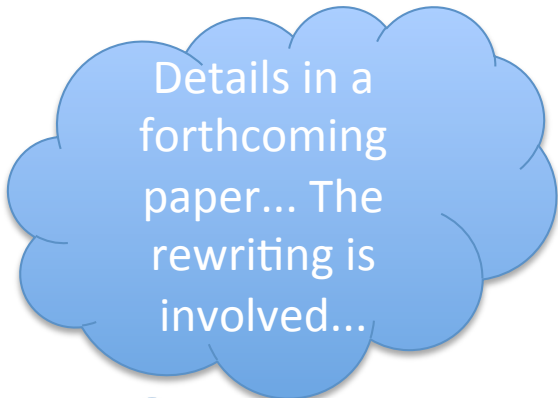
Breaking atomicity by rewriting

```
DELETE{?X :Regent}  
INSERT{?X :hasRegent ?Y}  
WHERE{?Y :signsDecreeInNameOf ?X}
```



```
DELETE{?X :Regent}  
INSERT{?X1 :hasRegent ?Y1}  
WHERE{ {?Y :signsDecreeInNameOf ?X}
```

```
OPTIONAL { ... bind ?X1 and ?Y1 "as needed" } }
```



Details in a
forthcoming
paper... The
rewriting is
involved...



Breaking atomicity using transactions

ALTERNATIVE: Split DELETE and INSERT into two queries.

Step 1: Prepare the bindings for insertions and deletions in a temporary graph.

```
DELETE{?Z a :Regent}
INSERT{?X :hasRegent ?Y}
WHERE{?Y :signsDecreeInNameOf ?X ... }
INSERT{?Z a :Regent_del,
?X :hasRegent_ins ?Y}
WHERE{?Y :signsDecreeInNameOf ?X...}
```

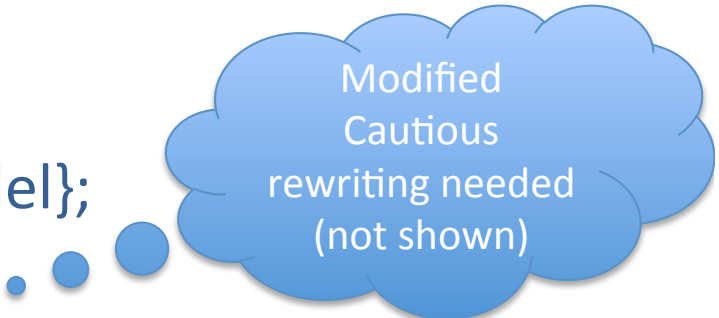
Step 2: Run insert and delete separately, but in a single transaction.

Step 2a:

```
DELETE{?Z a :Regent} WHERE {?Z a Regent_del};
```

Step 2b:

```
INSERT{?X :hasRegent ?Y} WHERE {?X :hasRegent_ins ?Y};
```



Modified
Cautious
rewriting needed
(not shown)


Plans to redraw map of France's regions described as 'butchery' as parliamentary debate opens

France's National Assembly begins delicate task of debating cutting the country's regions from 22 to as few as 13, as critics say it amounts to "butchery"

f 81 t p 0 in 0 s 81 Email



Plans to redraw map of France's regions described as 'butchery' as parliamentary debate opens

Dbpedia &
SPARQL Update
to the
rescue! 

Use case idea:
Use mappings to
update infoboxes
and track pages
that need updating.

- Import data from infoboxes using declarative mappings.
- Respecting additional taxonomy.

Marie Antoinette

Queen of France



Marie Antoinette with the Rose

Portrait by [Louise Élisabeth Vigée Le Brun](#), 1783.

Queen consort of France and Navarre

Tenure 10 May 1774 – 4 September 1791

Queen consort of the French

Tenure 4 September 1791 –
10 August 1792

Born 2 November 1755
[Hofburg Palace](#), [Vienna](#), [Austria](#),
[HRE](#)

Died 16 October 1793 (aged 37)
[Place de la Révolution](#), [Paris](#),
[France](#)

Burial 21 January 1815
[Saint Denis Basilica](#), [France](#)

Spouse [Louis XVI of France](#)

DBpedia Mapping Types:

- Page title corresponds to the subject URI.
- `PropertyMapping`: translate infobox property names to RDF predicate names.
- `IntermediateNodeMapping`: create auxiliary resources.
- `ConditionalMapping`: select concept and predicate names based on a condition.
- `GeocoordinateMapping`
- `DateIntervalMapping` split a “from-to” string.
- `TableMapping` (very rarely used).

Conditional Mapping: Example

```
{{ ConditionalMapping | cases =
  {{ Condition
    | templateProperty = наставка
    | operator = isSet
    | mapping = {{ TemplateMapping | mapToClass = OfficeHolder | mappings =
      {{ConstantMapping | ontologyProperty = gender | value = http://dbpedia.org/resource/Female}}}}}}
  {{ Condition
    | operator = otherwise
    | mapping = {{ TemplateMapping | mapToClass = OfficeHolder | mappings =
      {{ConstantMapping | ontologyProperty = gender | value = http://dbpedia.org/resource/Male}}}}}}

| defaultMappings =
  {{ PropertyMapping | templateProperty = име | ontologyProperty = foaf:name }}
  {{ PropertyMapping | templateProperty = роден-дата | ontologyProperty = birthDate }}
  ...
}}
```


Marie Antoinette
Queen of France



Marie Antoinette with the Rose
Portrait by Louise Élisabeth Vigée Le Brun, 1783.

Queen consort of France and Navarre
Tenure 10 May 1774 – 4 September 1791

Queen consort of the French
Tenure 4 September 1791 – 10 August 1792

Born 2 November 1755
Hofburg Palace, Vienna, Austria, HRE

Died 16 October 1793 (aged 37)
Place de la Révolution, Paris, France

Burial 21 January 1815
Saint Denis Basilica, France

Spouse Louis XVI of France

Use Case 1: Translating Updates

Spouse Louis XVI of France

Spouse ~~Louis XV of France~~

Questions/Problems:

- Does an edit lead to inconsistencies on other pages?
 - Is there a DBpedia update equivalent to this edit?
- If yes: under which semantics?


```
ON wikiPage=https://en.wikipedia.org/wiki/Caen
UPDATE InfoboxTemplate(French commune).region=NULL
WHERE InfoboxTemplate(French commune).region=[[Lower_Normandy]]
```

```
ON wikiPage=https://en.wikipedia.org/wiki/Caen
INSERT InfoboxTemplate(French commune).region=[[Normandy]]
```

translateDML

Match update and template properties

```
Mapping en:Infobox French commune
{{TemplateMapping
| mapToClass = Settlement
| mappings =
{{ConstantMapping | ontologyProperty = country | value = France }}
{{PropertyMapping | templateProperty = name | ontologyProperty = foaf:name }}
{{PropertyMapping | templateProperty = region | ontologyProperty = region }}
{{PropertyMapping | templateProperty = area km2 | ontologyProperty = area
| unit = squareKilometre }}
{{PropertyMapping | templateProperty = population | ontologyProperty =
populationTotal }}
{{GeocoordinatesMapping | latitude = latitude | longitude = longitude }}}}
```

Apply Mapping

```
{{Infobox French commune
| name = Caen
| region = Lower-Normandie
| department = Calvados
| longitude = -0.37
| latitude = 49.18
| area km2 = 25.70
| population = 108365}}
```

getTemplates('Caen')

```
DELETE { ?X :region :Upper_Normandy .
        ?Y :region :Lower_Normandy . }
INSERT { ?X :region :Normandy . ?Y :region :Normandy }
WHERE
{ { ?X :region :Upper_Normandy . } UNION
  { ?Y :region :Lower_Normandy . }
}
```

Dbpedia SPARQL endpoint

eval(WHERE)

```
DELETE { :Rouen :region :Upper_Normandy .
:Le_Havre :region :Upper_Normandy . ...
:Caen :region :Lower_Normandy .
:Bayeux :region :Lower_Normandy . ... }
INSERT { :Rouen :region Normandy .
:Le_Havre :region :Normandy . ...
:Caen :region :Normandy .
:Bayeux :region :Normandy . ... }
```

Separate updates

```
DELETE { :Rouen :region :Upper_Normandy .
:Le_Havre :region :Upper_Normandy . ...
:Caen :region :Lower_Normandy .
:Bayeux :region :Lower_Normandy . ... }
```

```
INSERT { :Rouen :region :Normandy .
:Le_Havre :region :Normandy . ...
:Caen :region :Normandy .
:Bayeux :region :Normandy . ... }
```

Conclusions

- First steps to close the gap left by the current standards
(SPARQL1.1 Update vs. SPARQL1.1 Entailment Regimes)
- We looked into **materialized-preserving** and also reduced-preserving semantics
 - Focus on implementability in off-the-shelf SPARQL engines with "on-board" means
 - Seemingly no “one-size fits all” semantics
 - Even for RDFS only
 - Bad news: adding OWL features makes things probably worse!
E.g. adding class inconsistency handling adds even more degrees of freedom.
 - Non-intuitive corner cases in **each** semantics
 - depends on use case?
 - Which Semantics works in practice for which use case?
 - DBpedia/wikipedia updates as "real use case" is only a sketch so far.
 - Which postulates are satisfied by which semantics?
 - Yet Other possible semantics?
 - All our semantics so far are formula-based, can a model-based semantics also be implemented in a triple store (e.g. by storing models in different named graphs?)
- Take-home: SPARQL 1.1 Update, i.e. pairing DELETE and INSERT templates with a common WHERE clause (BGP matching) imposes a non-trivial challenge!

Prototype & Evaluation

- A prototype in Java using Jena API implementing the proposed update semantics
 - <http://dbai.tuwien.ac.at/user/ahmeti/sparqlupdate-rewriter/index.html>

Update + 14 LUBM Select queries using LUBM(15) dataset

