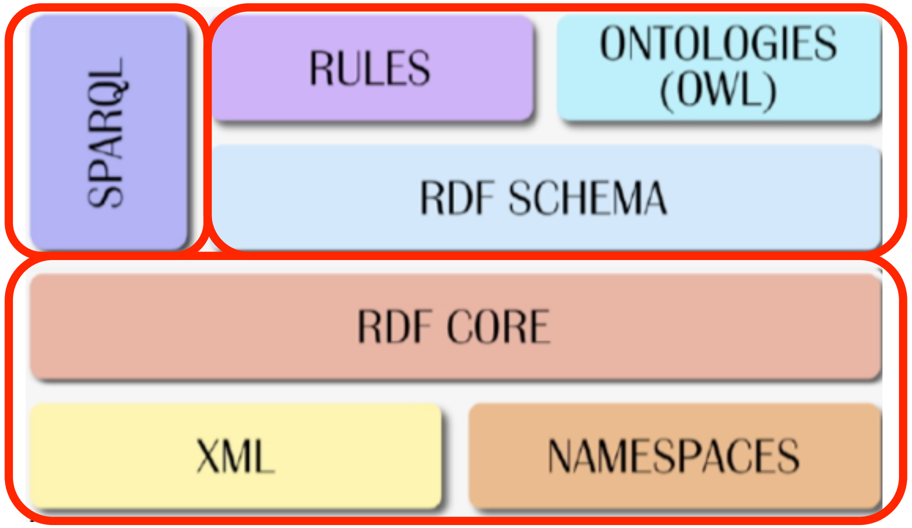
Semantic Web Technologies: From Theory to Practice

Axel Polleres
Digital Enterprise Research Institute, National University of Ireland, Galway

Habilitationskolloqium, TU Vienna, 07/03/2011

The Semantic Web in W3C's view:

3. Shall allow us to ask **structured queries** on the Web



2. Shall allow us to describe the structure of information in machine readable form: **RDFS+OWL+RIF**

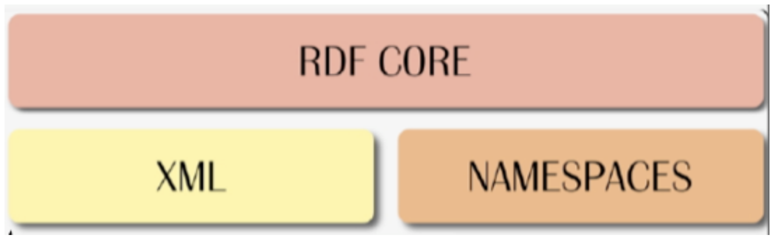
1. Shall allow us to publish structured information on the Web: **XML+RDF**

Focus of my Habilitation:

- Which theory do these Semantic Web standards base on?
- What's missing? (= Do these standards work together)

1. Structured Data on the Web

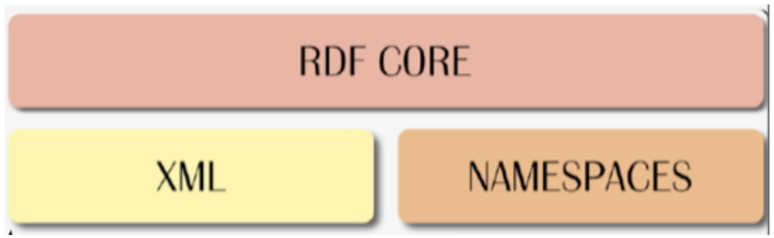
"Prof. Seidler is the new Rector of TU Vienna"



1. Structured Data on the Web

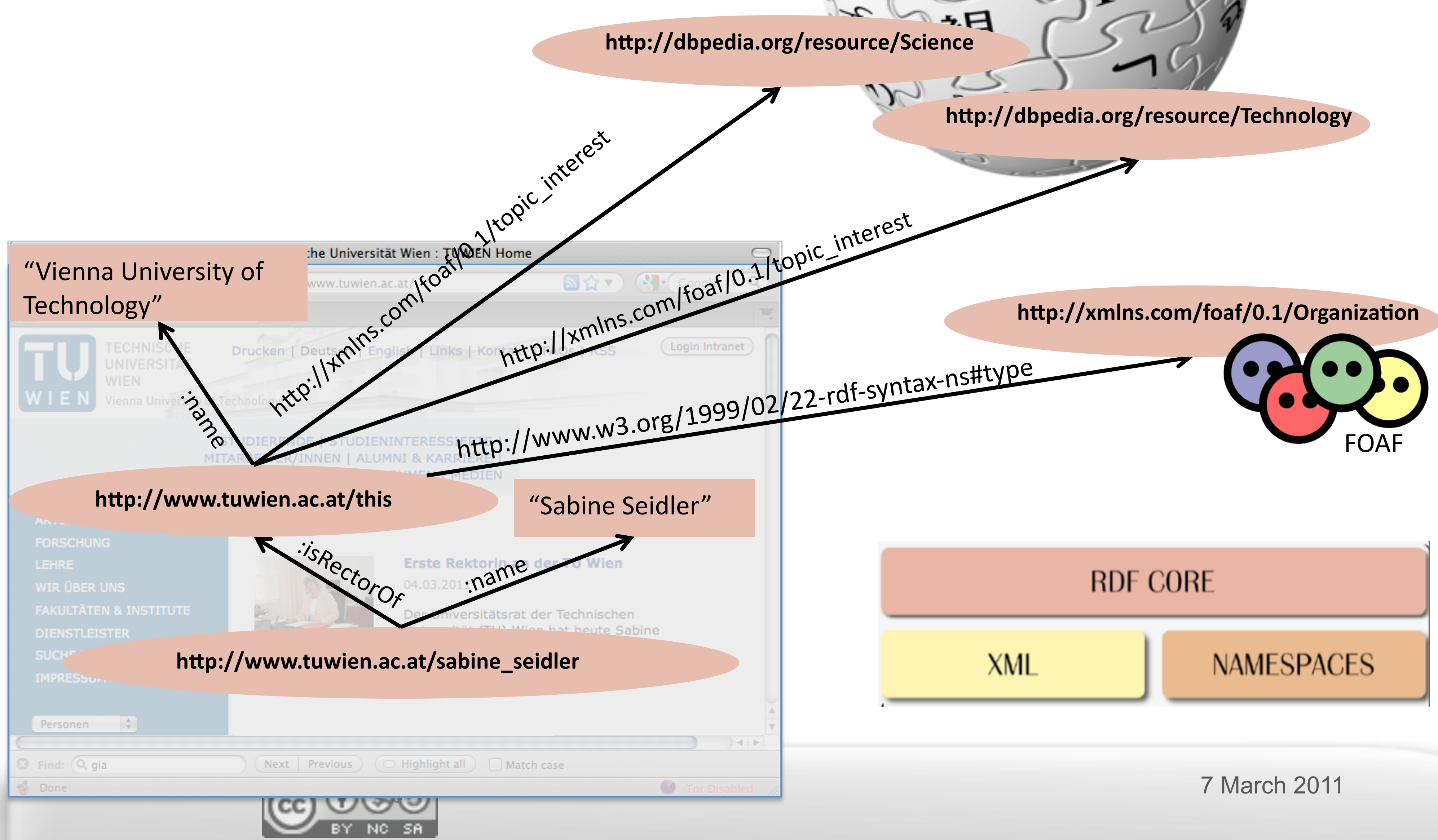
"Prof. Seidler is the new Rector of TU Vienna"

```
<organisation xmlns="http://www.tuwien.ac.at/ns/">
  <name>Vienna University of Technology</name>
  <employees>
    < person id="person1" gender="female">
    <name>Sabine Seidler</name>
    <title>rector</title>
  </person>
  <employees>
  ...
</organisation>
```



1. Structured Data on the Web

"Prof. Seidler is the new Rector of TU Vienna"



RDF is knowledge representation

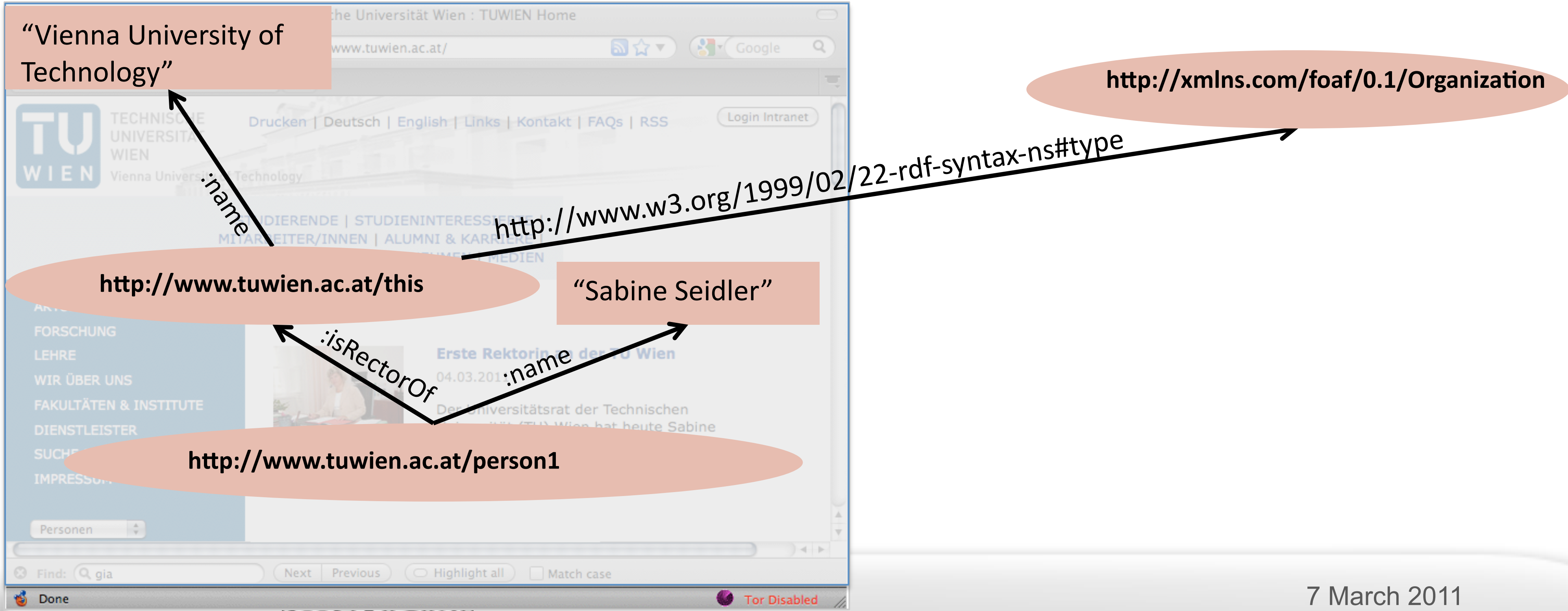
Each RDF Graph is a set of Triples (Subject, Predicate, Object)

<http://www.tuwien.ac.at/person1> :name ”Sabine Seidler" .

<http://www.tuwien.ac.at/person1> :isRectorOf <http://www.tuwien.ac.at/this>.

<http://www.tuwien.ac.at/this> :name “Vienna University Of Technology”.

<http://www.tuwien.ac.at/this> rdf:type <http://xmlns.com/foaf/0.1/Organization>.



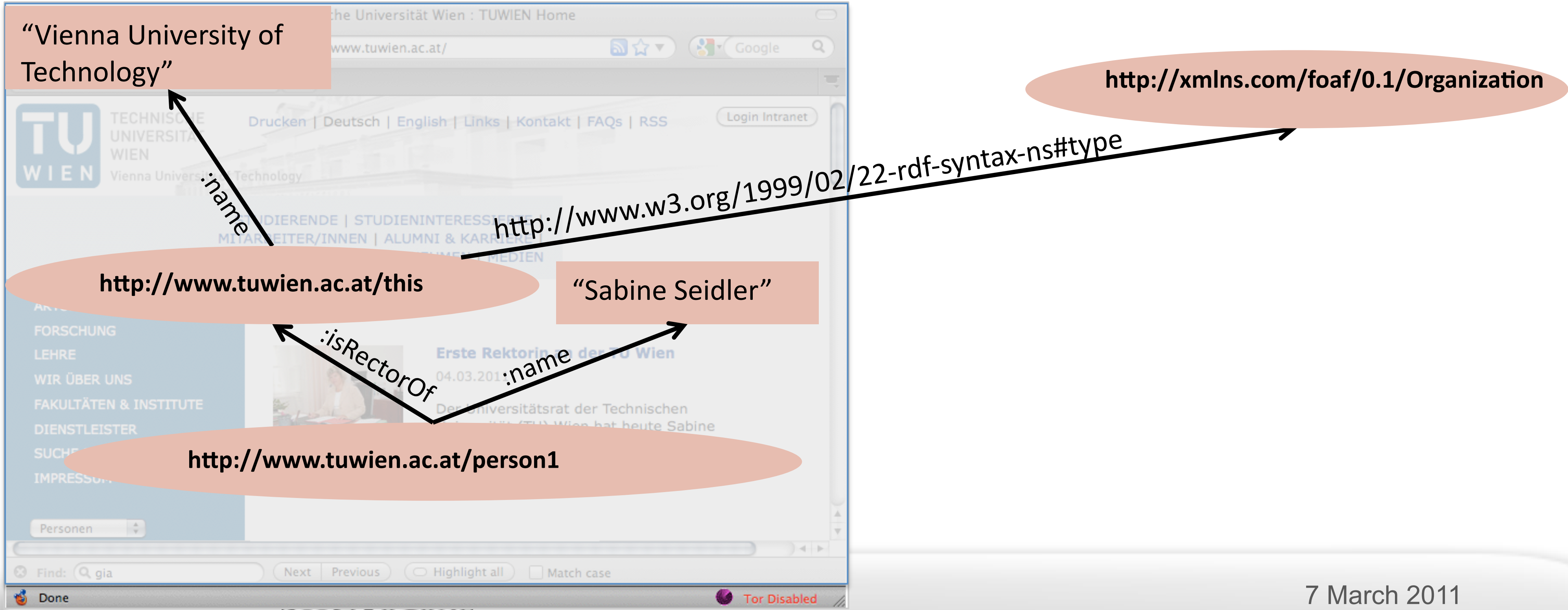
Each RDF Graph is a set of Triples (Subject, Predicate, Object)

RDF is knowledge representation

name(person1, "Sabine Seidler")
∧
isRectorOf(person1, tuwien)
∧
Name(tuwien, "Vienna University of Technology")
∧
Organisation(tuwien)

RDF+RDF Schema can be embedded in ***FOL*** [deBruijn et al. 2005] …

…or ***Datalog*** [deBruijn et al. 2007] [Ianni et al. 2009]



2. RDF can be described in terms of Ontologies and Rules ➔ *allows Reasoning!*

"Persons ***work with Organisations****"*
"The ***rector of*** *a university* ***works with*** *that university"*

➔ RDF Schema (RDFS)

"Every university ***has at most one rector****"*

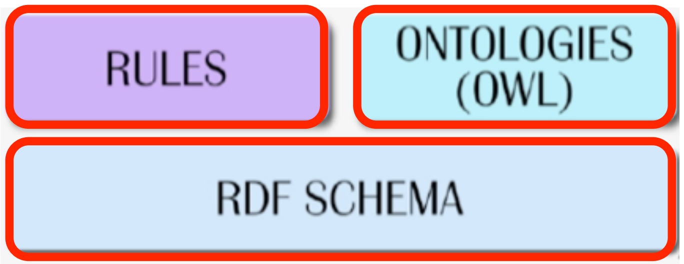
➔ Web Ont. Lang. (OWL)

"Persons who ***worksWith*** *a university and* ***didn't study there (CWA)*** *are* ***influx Personnel*** *of that univ."*

➔ Rule Interchange Format (RIF)

:worksWith rdfs:range :Organisation.
:isRectorOf rdfs:subPropertyOf :worksWith.
:isRectorOf rdf:type owl:inverseFunctionalProperty.

{ ?P rdf:type :InfluxPersonnel } **IF**
{?P :worksWith ?U } **AND NOT EXISTS** {?P :studiedAt ?U}



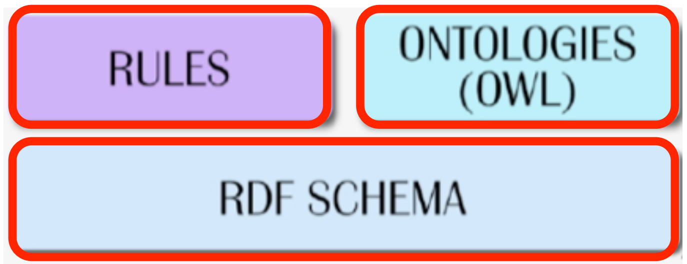
2. RDF can be described in terms of Ontologies and Rules → *allows Reasoning!*

isRectorOf(person1, tuwien).

studiedAt(person9225749, tuwien).

$$\forall X \forall Y (worksWith(Y,X) \rightarrow Organisation(X))$$
$$\forall X \forall Y (isRectorOf(Y,X) \rightarrow worksWith(X,Y))$$
$$\forall X \forall Y \forall Z (isRectorOf(X,Z) \land isRectorOf(Y,Z) \rightarrow X = Y)$$

InfluxPersonnel(P) ← *worksWith(P,U),* ***not*** *studiedAt(P,U).*



worksWith(person1, tuwien).

Organisation(tuwien).

InfluxPersonnel(person1).

2. Structured queries over Web data

- SPARQL = "SQL look-and-feel query language for the Web"
- allows us to ask structured queries such as:

"Persons who work for a technology organization"

```
SELECT ?P
 { ?P rdf:type :Person.
   ?P :worksWith ?O .
   ?O :topic_interest dbpedia:Technology
 }
```

Unions of conjunctive queries, but also advanced features such as outer joins (OPTIONAL), value filtering, etc.

2. Structured queries over Web data

SPARQL

- SPARQL = "SQL look-and-feel query language for the Web"
- allows us to ask structured queries such as:

"All Persons and optionally the organization they work for"

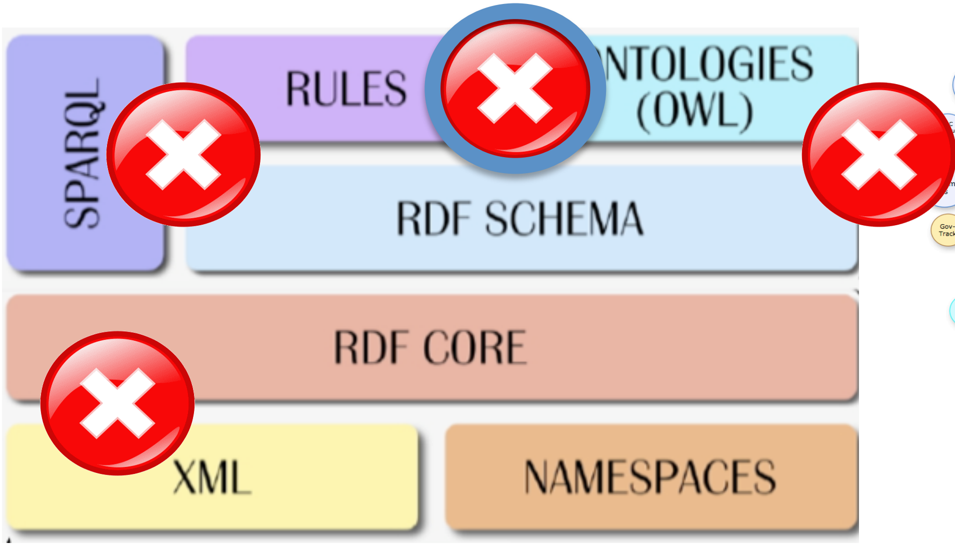
```
SELECT ?P ?O
 {
   ?P rdf:type :Person.
   OPTIONAL  {?P :worksWith ?O .}
 }
```

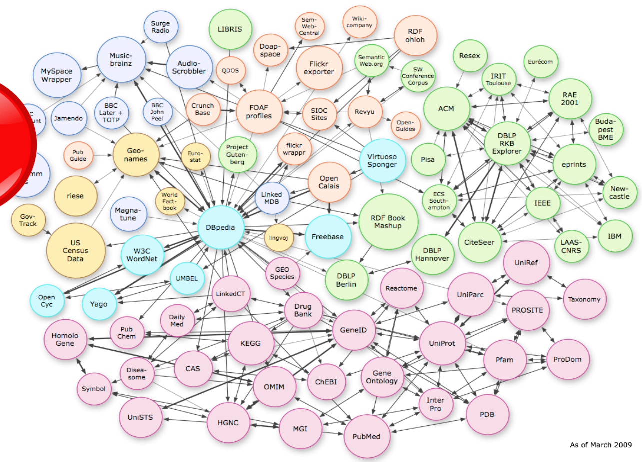
Unions of conjunctive queries, but also advanced features such as outer joins (OPTIONAL), value filtering, etc.

P	O
p1	tuwien
p9225749	

How do the standards interplay?

- Challenges:
  1) Combining Rules & Ontologies
  2) Querying Rules & Ontologies
  3) Data on the Web is NOT clean/consistent!
  4) Querying XML & RDF





1) Combining OWL Ontologies & nonmomotonic Rules

KB = (T ,P)

$\forall X \forall Y (worksWith(Y,X) \rightarrow Organisation(X))$
$\forall X \forall Y (isRectorOf(Y,X) \rightarrow worksWith(X,Y))$
$\forall X \forall Y \forall Z (isRectorOf(X,Z) \wedge isRectorOf(Y,Z) \rightarrow X = Y)$

$Personnel(P) \leftarrow worksWith(P,U).$

- Combination of LP (even function-free Horn) + relatively simple DLs already leads to undecidability [Levy, Rousset 1998] …

- *BUT:* semantics is clear… embedding into classical first-order logic (**FOL**)

1) Combining OWL Ontologies & nonmomotonic Rules

KB = (T ,P)

$$\forall X \forall Y (worksWith(Y,X) \rightarrow Organisation(X))$$
$$\forall X \forall Y (isRectorOf(Y,X) \rightarrow worksWith(X,Y))$$
$$\forall X \forall Y \forall Z (isRectorOf(X,Z) \wedge isRectorOf(Y,Z) \rightarrow X = Y)$$

?

InfluxPersonnel(P) ← *worksWith(P,U),* ***not*** *studiedAt(P,U).*

– Combination of LP (even function-free Horn) + relatively simple DLs already leads to undecidability [Levy, Rousset 1998] …

– *BUT:* semantics is clear… embedding into classical first-order logic (**FOL**)

- **Open question:**

– reasonable underlying logics for combinations of DLs/FOL and non-monotonic LPs?

Can't be embedded in FOL, because non-mon. logic programs under ASP rely on a non-classical logic!

Starting Point: Hybrid Knowledge Bases & NM-models

$$\mathcal{K} = (\mathcal{T}, \mathcal{P})$$

- **Hyrid KB** approaches rely on (variants of) the Answer Set Semantics. [Rosati, 2005/2005b/2006, Heymans, et al. 2006]
  - Define so called NM-Models for syntactically limited programs/FOL theories
  - All give a modular definition of models by projection+reduct:
    1. Ground $\mathcal{P}$ over the constants in KB: gr($\mathcal{P}$)
    2. For all FO-Models M over the signature of $\mathcal{T}$ create the "*projection*" $gr(\mathcal{P})|_M$ wrt. M.
    3. *Evaluate* all answer sets of $gr(\mathcal{P})|_M$

NM-model an example:

$\forall X \forall Y (isRectorOf(Y, X) \rightarrow worksWith(X, Y))$

$M_1 = \{isRectorOf(p1, tuwien), worksWith(p1, tuwien)\}$
$M_2 = \{\neg isRectorOf(p1, tuwien), worksWith(p1, tuwien)\}$
$M_3 = \{\neg isRectorOf(p1, tuwien), \neg worksWith(p1, tuwien)\}$

studiedAt(p9225749,tuwien) .
isRectorOf(p1, tuwien) .
InfluxPersonnelTU(X) ← worksWith(X,U), ***not*** *studiedAt(X,U)*

Is *p1* a member of the class *InfluxPersonnel*?

NM-models **main idea**:
The *projection* gr($\mathcal{P}$)|$_{Mi}$ "***applies***" each model classical M_1, M_2, M_3 to the logic program, before stable models are computed.

- *For* $M_{2,3}$, gr($\mathcal{P}$)|$_{Mi}$ *contains unsatisfiable constraint,* ← . *so no stable model.*
- *For* M_1, gr($\mathcal{P}$)|$_{Mi}$ *contains the ground rule instance:*

InfluxPersonnelTU(p1) ← ***not*** *studiedAt(p1,tuwien).*

•*Thus, InfluxPersonnelTU(p1)* remains in the only stable model, i.e., in the NM-model

Alternative – Model Theoretic approach

Back to our question for a non-classical logic which covers this. . .

- ***Equilibrium logic*** [Pearce, 1997] generalizes stable model semantics and answer set semantics for logic programs to arbitrary **propositional** theories.

We defined a **first-order version: Quantified Eqilibrium Logic (QEL)**

- Model theory based on Kripke semantics for intuitionistic logic.
- Special Kripke models with two worlds **H ≤ T**, thus the Kripke models are also called "Here-and-there" models, each world is a FO-model.
- Quantified **Equilibrium** models are models ⟨H, T⟩ where
  - H=T (**total**) and
  - there exists no QHT ⟨H', T⟩ model such that H' ⊊ H (**minimality**).

How to embed Hybrid KBs in QEL?

- We can embed both $\mathcal{T}$ the $\mathcal{P}$ in QEL…
- **Question:** For $\mathcal{K}$ = ($\mathcal{T}$,$\mathcal{P}$) how can we make the predicates from $\mathcal{T}$ "behave classical" within QEL?
- **Observation:** If LEM is added to intuitionistic logic, then intuitionistic logic collapses to classical logic.
- **Idea**: Add LEM only for the classical predicates:

Definition:

Given a hybrid KB $\mathcal{K} = (\mathcal{T}, \mathcal{P})$ we call $\mathcal{T} \cup st(\mathcal{T}) \cup \mathcal{P}$ the stable closure of $\mathcal{K}$, where $st(\mathcal{T}) = \{\forall x(p(x) \vee \neg p(x)) : p \in \mathcal{L}_\mathcal{T}\}$.

Example?

- Transform $\mathcal{K} = (\mathcal{T},\mathcal{P})$ into a first-order theory…

$\mathcal{T}$ $\forall X \forall Y (isRectorOf(Y,X) \rightarrow worksWith(X,Y))$

$st(\mathcal{T})$ $\forall X \forall Y (isRectorOf(X,Y) \vee \neg isRectorOf(X,Y))$

$\forall X \forall Y (worksWith(X,Y) \vee \neg worksWith(X,Y))$

$\mathcal{P}$ *studiedAt(p9225749,tuwien)*

isRectorOf(p1, tuwien)

$\forall X \forall U (worksWith(X,U) \wedge \neg studiedAt(X,U) \rightarrow InfluxPersonnelTU(X))$

- … and interpret it in QEL.

- Now… QEL Models of the *stable closure* coincide exactly with the NM-models from [Rosati,2005/2005b/2006] !!!

Main results:

Theorem

Let $\mathcal{K} = (\mathcal{T}, \mathcal{P})$ be a hybrid knowledge base. Let $\mathcal{M} = \langle U, T, T \rangle$ be a total here-and-there model of the stable closure of $\mathcal{K}$. Then $\mathcal{M}$ is an equilibrium model if and only if it is an NM-model of $\mathcal{K}$.

- i.e., stable closure in QEL can replace the "modular" semantics definition of NM-models !!
- Added value: Definition of **strong equivalence** for **Hybrid KBs**:

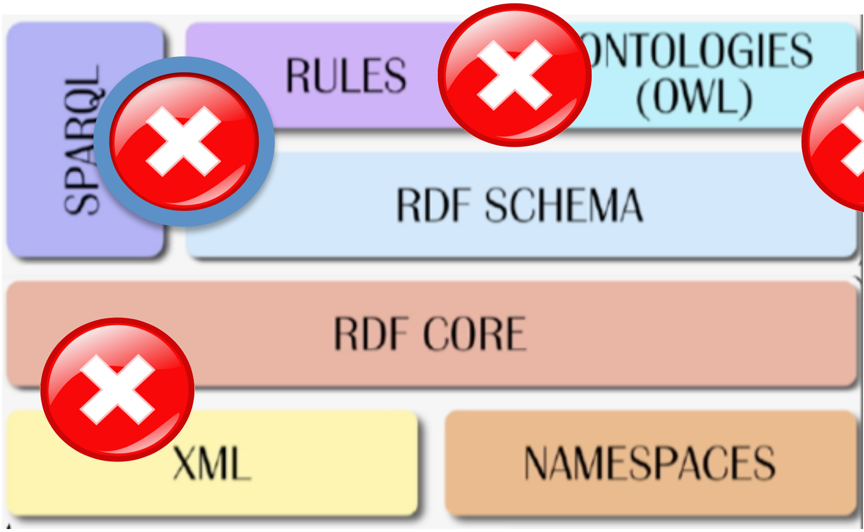
Definition

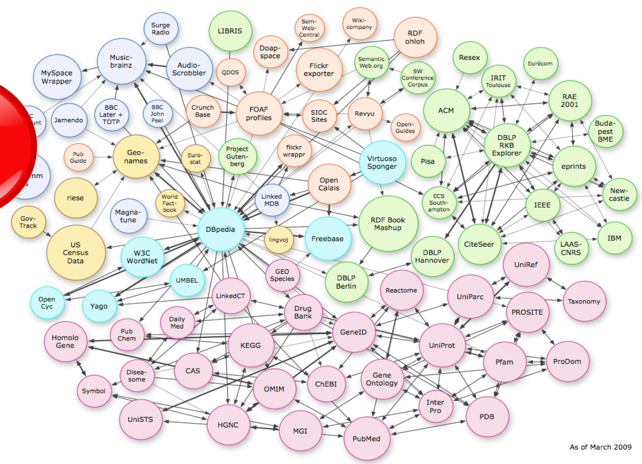
Two hybrid KBs $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{P}_1)$ and $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{P}_2)$ over same structural language are called *strongly equivalent* if for any theory $\mathcal{T} = (\mathcal{T}, \mathcal{P})$, $(\mathcal{T}_1 \cup \mathcal{T}, \mathcal{P}_1 \cup \mathcal{P})$ and $(\mathcal{T}_2 \cup \mathcal{T}, \mathcal{P}_2 \cup \mathcal{P})$ have the same NM-models.

- $\mathcal{K}_1$ and $\mathcal{K}_2$ are strongly equivalent if $\mathcal{T}_1$ and $\mathcal{T}_2$ are classically equivalent and $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent in $\mathbf{QHT}$.
- $\mathcal{K}_1$ and $\mathcal{K}_2$ are not strongly equivalent if $\mathcal{T}_1 \cup \mathcal{P}_1$ and $\mathcal{T}_2 \cup \mathcal{P}_2$ are not equivalent in classical logic.
- $(\mathcal{T}_1, \mathcal{P})$ and $(\mathcal{T}_2, \mathcal{P})$ are strongly equivalent if $\mathcal{T}_1$ and $\mathcal{T}_2$ are classically equivalent.

How do the standards interplay?

- Challenges:
  1) Combining Rules & Ontologies
  2) Querying Rules & Ontologies
  3) Data on the Web is NOT clean/consistent!
  4) Querying XML & RDF





2007…

- SPARQL's semantics under discussion [Perez et al. 2006], no standard recommendation yet
- Also: SPARQL is for RDF only…
- unclear how to combine SPARQL with Rules & OWL
- **Observations**:
  - A good bit of OWL and RDFS can be evaluated by simple Datalog rules (DLP, pD*, later OWL2RL), most engines do that
  - A good bit of SQL is reducible to Datalog as well…

…Can we use deductive databases techniques to answer SPARQL queries over RDFS?

SQL to Datalog…

- A large part of SQL has a natural correspondence to Datalog … e.g.

Example: Two tables containing adresses and universities

```
Addr(Name, Street, City, Telephone)
Universtiy(Name, City)
```

“Give me all names of people living in a city with a University”

Easy!

```
SELECT Addr.name FROM Addr,University
 WHERE Addr.City = University.City;
```

```
answer(AddrName) :- Addr(AddrName, Street, City, Tel).
                      City(CityName, City).

?- answer(Name).
```

SPARQL to Datalog…

- Should be doable for SPARQL as well…
  benefit: can be combined with inference rules emulating RDFS, OWL!

Recall from earlier: *"Persons who work for a technology organization"*

```
SELECT ?P
{ ?P rdf:type :person.
  ?P :worksWith ?O .
   ?O :topic_interest :technology
}
```

Looks even easier!

```
answer(P) :-    RDF(P,rdf:type,person),
                RDF(P,worksWith,O),
                RDF(O,topic_interest, technology).
RDF(S,Q,O) :-  RDF(S,P,O), RDF(P, rdfs:subPropertyOf, Q).
```

SPARQL to Datalog…

- More tricky bits, e.g. OPTIONAL

Recall from earlier: *"All Persons and optionally the organization they work for"*

```
SELECT ?P ?O
{
?P rdf:type :person.
  OPTIONAL {?P :worksWith ?O .}
}
```

```
answer(P, O)    :- RDF(P,rdf:type,person), RDF(P,worksWith,O).
answer(P, null) :- RDF(P,rdf:type,person), not answer1(P).
answer1(P)      :- RDF(P,worksWith,O).
```

i.e., SPARQL's Outer Join (OPTIONAL) can be emulated with null and negation as failure.

SPARQL to Datalog…

:p1	rdf:type	:Person .
:p1	:worksWith	:tuwien .
:p9225749	rdf:type	:Person .
:tuwien	:topic_interest	:technology.

ATTENTION: non-SQL compatible JOIN semantics for null:

null joins with anything in SPARQL!

```
SELECT ?P ?O
{
{ ?P rdf:type :person.
 OPTIONAL  {?P :worksWith ?O .}
{?O :topic_interest :technology}
}
```

P	O
p1	tuwien
p9225749	null

⋈

O
tuwien

=

P	O
p1	tuwien
p9225749	**tuwien**

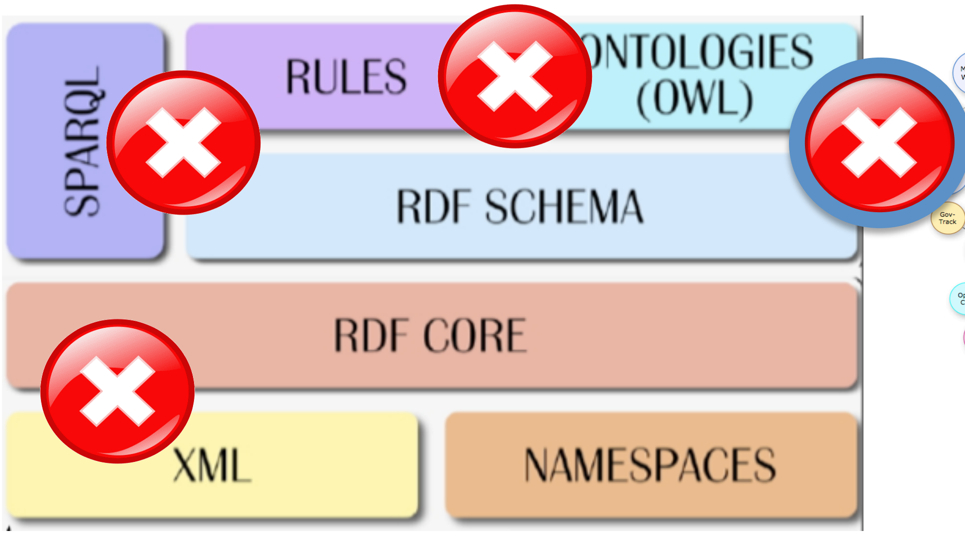
Translation to non-recursive Datalog^{not} needs slight adaption, but still polynomial, details in **[WWW2007]**

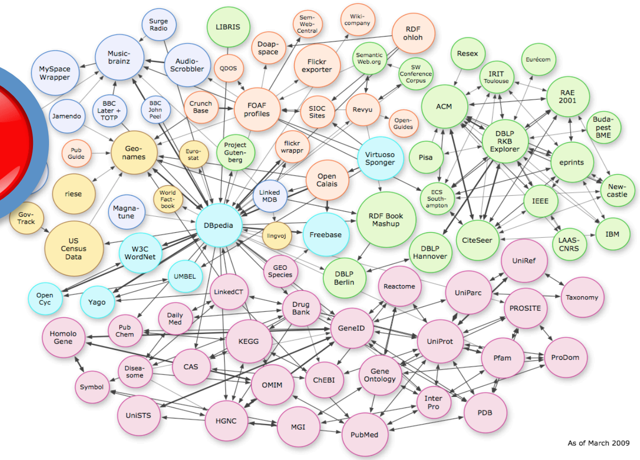
Main results:

- All SPARQL queries can be translated to non-recursive Datalog^{not} **[WWW2007]**
  - Confirming feasibility by PSPACE complexity results from [Perez et al. 2006]
  - [Angles, Gutierrez, 2008] showed reverse translation, i.e. expressivity of SPARQL is precisely the same as non-recursive Datalog^{not}
- Follow-up works included: proposing SPARQL for complex mappings between ontologies **[ODBASE2007]**:
  - Based again on reduction to ASP
  - Adding expressive features "for free**"** **(aggregates, built-ins)**
    - → **now on the agenda of SPARQL1.1**
  - Similar proposal, based on WFS by [Shenk, Staab, 2008]
- Implementation and proposed semantics for SPARQL + OWL/Rules **[ISWC2009]**
  - Implementation based on dlv-DB and earlier results from [Ianni et al. 2007,2009]
  - Basis for **SPARQL1.1/RIF** semantics

How do the standards interplay?

- Challenges:
  1) Combining Rules & Ontologies
  2) Querying Rules & Ontologies
  3) Data on the Web is NOT clean/consistent!
  4) Querying XML & RDF





SAOR - Reasoning for SWSE

http://swse.deri.org/

- We want explicit RDF data **plus** OWL inferred data in the search results!
- Our approach:

  SAOR – **S**calable **A**uthoritative **O**WL **R**easoning

Idea

- Apply a tailored subset of OWL2RL reasoning
- Forward-chaining rule based approach

- Reduced output statements for the SWSE use case…
  - Must be *scalable*, must be *reasonable, can't afford to index too much.*
- … incomplete w.r.t. OWL **BY DESIGN!**
  - **SCALABLE:** Tailored ruleset
    - file-scan processing
    - avoid joins
  - **AUTHORITATIVE:** Avoid Non-Authoritative inference
    ("hijacking", "non-standard vocabulary use")

Scalable Reasoning

- **Scan 1:**

  Scan all data (1.1b statements), ***separate T-Box statements***, load T-Box statements (8.5m) into memory, perform authoritative analysis.

- **Scan 2:**

  Scan all data and join all statements with in-memory T-Box .

  – Only works for inference rules with 0-1 A-Box patterns*
  – Inference rules only infer A-Box statements - no T-Box expansion by inference.

  → Needs “tailored” ruleset

* We also describe (necessarily less scalable) techniques for rules that need A-Box joins in the paper.

Rules Applied: We focus on rules that don't need A-Box joins

Tailored version of [ter Horst, 2005] in the spirit of OWL2RL

$\mathcal{G}1$: ONE A-BOX PATTERN IN ANTECEDENT

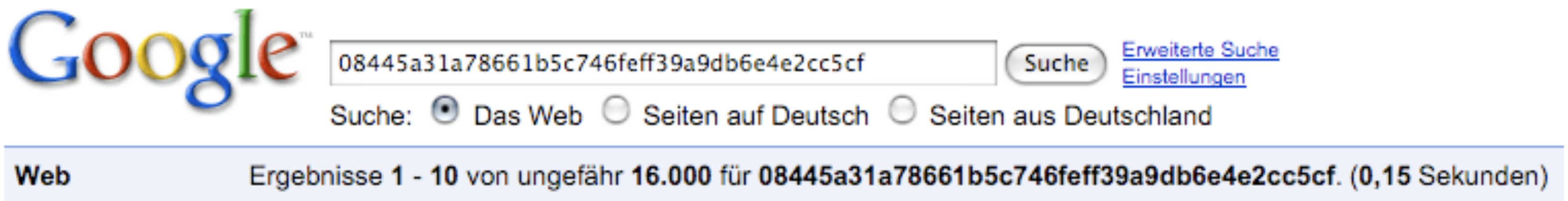
?x rdf:type ?C . ← **?P** rdfs:domain ?C . ?x ?P ?y .
?y rdf:type ?C . ← **?P** rdfs:range ?C . ?x ?P ?y .
?x ?Q ?y . ← **?P** rdfs:subPropertyOf ?Q . ?x ?P ?y .
?x rdf:type ?D . ← **?C** rdfs:subClassOf ?D . ?x rdf:type ?C .
?y ?P ?x . ← **?P** rdf:type :SymmetricProperty . ?x ?P ?y .
?y ?Q ?x . ← **?P** :inverseOf ?Q . ?x ?P ?y .
?y ?P ?x . ← ?P :inverseOf **?Q** . ?x ?Q ?y .
?x rdf:type ?D .← **?C** :equivalentClass ?D . ?x rdf:type ?C .
?x rdf:type ?C . ← ?C :equivalentClass **?D** . ?x rdf:type ?D .
?y ?Q ?x . ← **?P** :equivalentProperty ?Q . ?x ?P ?y .
?y ?P ?x . ← ?P :equivalentProperty **?Q** . ?x ?Q ?y .
?x rdf:type ?C . ← ?C :hasValue ?y ; :onProperty **?P** . ?x ?P ?y .
?x ?P ?y . ← ?C :hasValue ?y ; :onProperty ?P . ?x rdf:type ?C .
?x rdf:type ?C . ← ?C :unionOf (?C_1...**?C_i**...?C_n) . ?x rdf:type ?C_i .
?x rdf:type ?C . ← ?C :minCardinality 1 ; :onProperty **?P** . ?x ?P ?y .
?x rdf:type ?C_1, ..., ?C_n . ← ?C :intersectionOf (?C_1 ... ?C_n) . ?x rdf:type ?C .
?x rdf:type ?C . ← ?C :intersectionOf (**?C_1**) . ?x rdf:type ?C_1 .

Underlined patterns are from the T-Box, non-underlined are "Data" or A-Box triples.
Note that these rules cover RDFS reasoning.

Good “excuses” to avoid $\mathcal{G}2$ rules

- The obvious:
  - $\mathcal{G}2$ rules would need joins with inferred triples, i.e. to trigger restart of file-scan
- The interesting one:
  - Take for instance owl:inverseFunctionalProperty rule:

    $?x\ :sameAs\ ?y\ .\ \leftarrow\ \underline{\mathbf{?P}\ a\ :InverseFunctionalProperty\ .}\ ?x\ ?P\ ?z\ .\ ?y\ ?P\ ?z\ .$
  - Maybe not such a good idea on real Web data



sha1 hashsum of empty string…. Common value for foaf:mbox_sha1

Web Tolerance: Authoritative Reasoning

- We check authority (on the T-Box statements only) to make inferences!

- Document **D** authoritative for class/property **X** iff:
  - De-referenced URI of **X** coincides with or redirects to **D**
  - FOAF ontology authoritative for `foaf:Person` ✓
  - MY spec not authoritative for `foaf:Person` ✗

- **Borrowing from the idea of DL to separate T-Box and A-Box we enable authority checking by so called *split-rules* :**

  - *Split-rule*: Antecedent divided in **T-Box** and **A-Box** statements.
  - *Split-rule Application:* At least one of the **A-Box**/**T-Box** join variables needs to be spoken about authoritatively, for the rule to fire.

`?s rdf:type ?d . ← ?c rdfs:subClassOf ?d . ?s rdf:type ?c .`

Web Tolerance: Authoritative Reasoning

- **Example:**

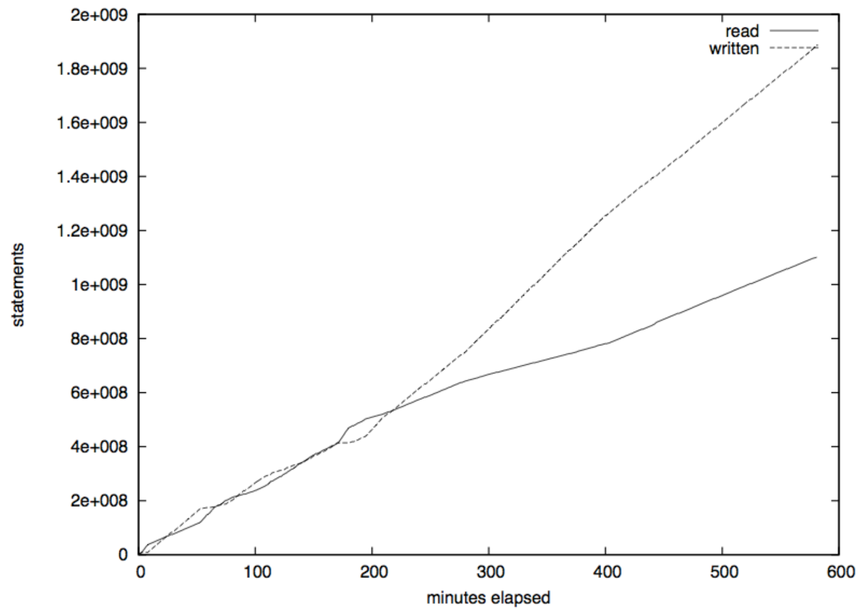
```
?s rdf:type ?d .   :-  ?c rdfs:subClassOf ?d  . ?s rdf:type ?c .
```

- Only allow extension in authoritative documents
  - `my:Person rdfs:subClassOf foaf:Person .` (MY spec) ✓
- BUT: Reduce obscure memberships
  - `foaf:Person rdfs:subClassOf my:Person .` (MY spec) ✘

Our In-memory T-Box only stores only statements that are authoritative for rule execution.

Main results

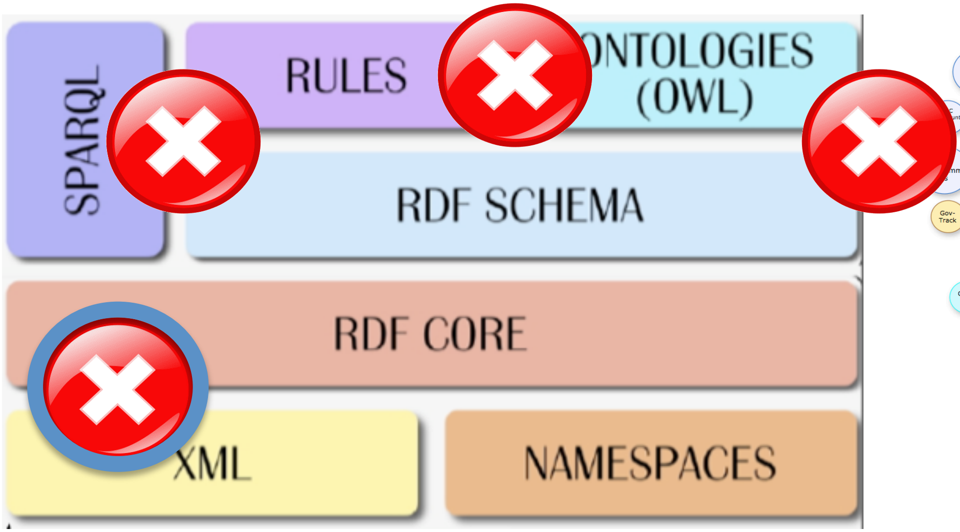
- Current experiments:
  - 1b of real Web crawl Linked Data data triples (only *G1 rules)*
  - 6hrs on one machine commodity hardware, inferred around 2b triples

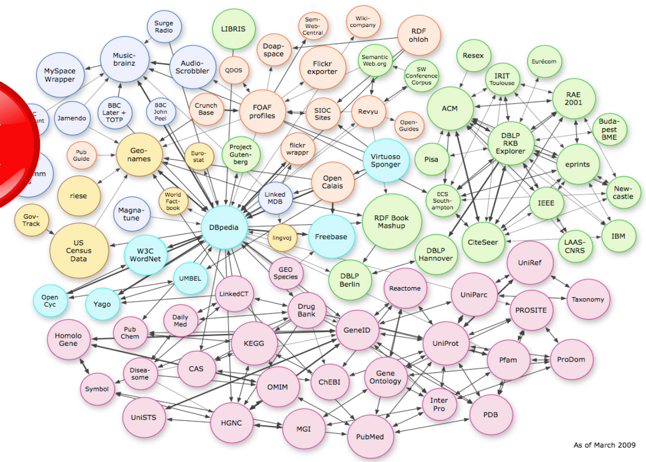


  - Including *G2, G3* rules over a smaller dataset (up to ~100M statements)
- *G1* rules Approach has been taken up can be easily distributed [Weaver&Hendler,2009], [Urbani etal.,2010]
  - but: no Web Data, no split rules!
- Several extensions e.g. [ISWC2010], PhD thesis defended [Hogan 2010]

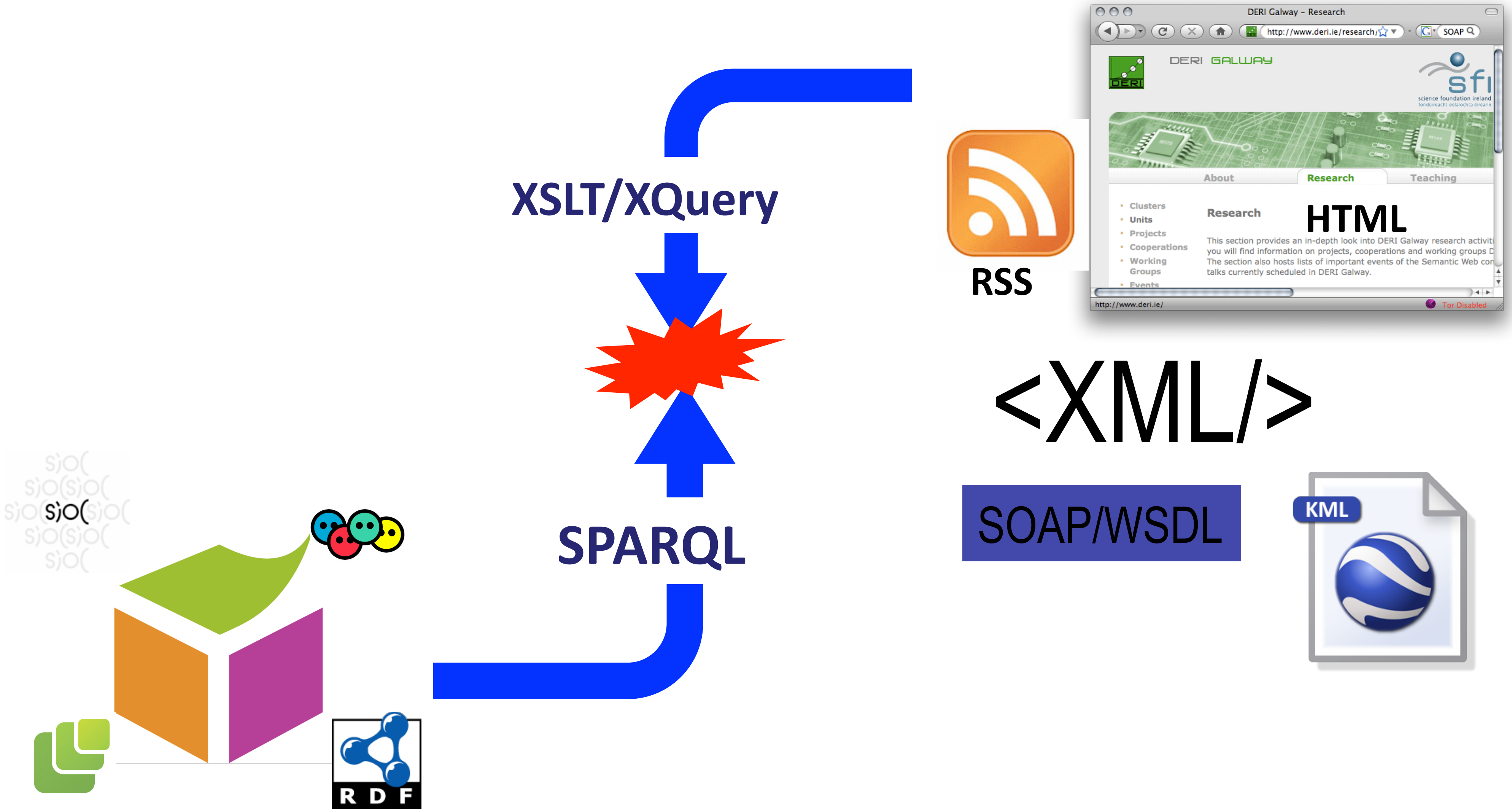
How do the standards interplay?

- Challenges:
  1) Combining Rules & Ontologies
  2) Querying Rules & Ontologies
  3) Data on the Web is NOT clean/consistent!
  4) Querying XML & RDF

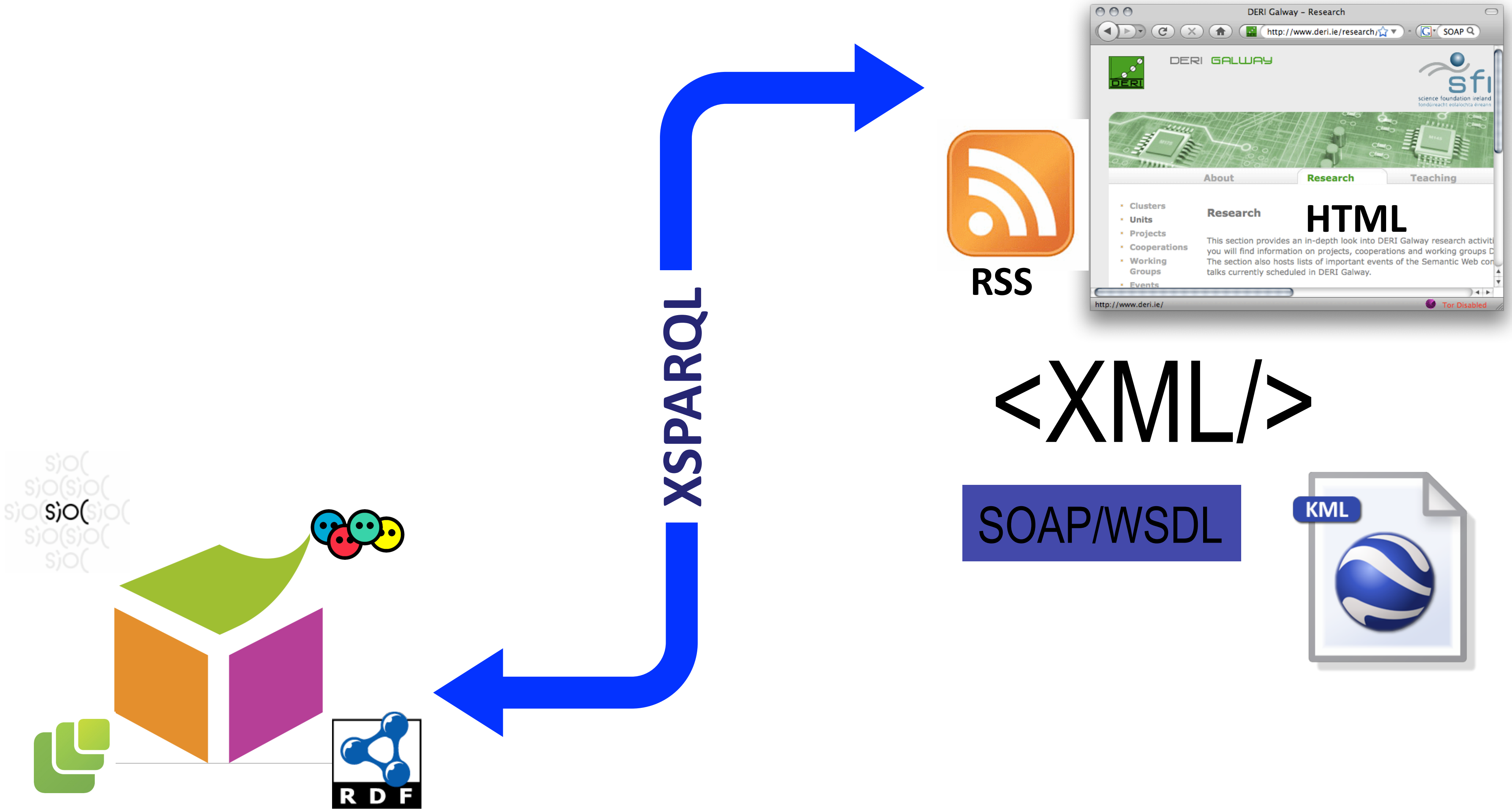




General motivation:



General motivation:



Example:

Lowering

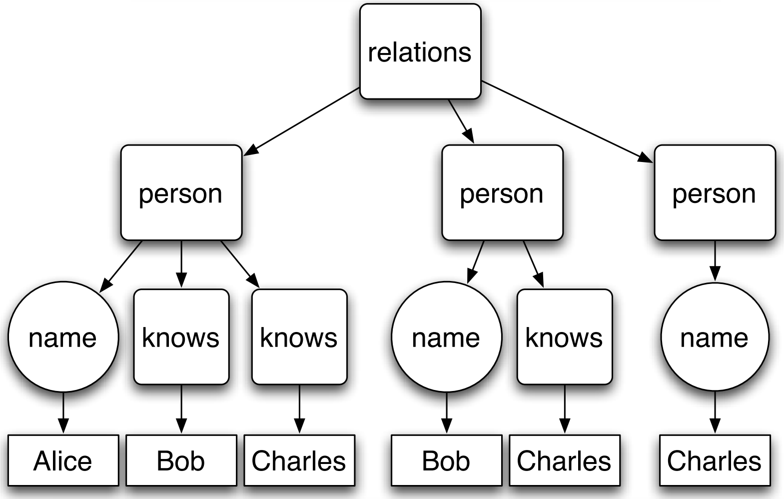
relations.xml

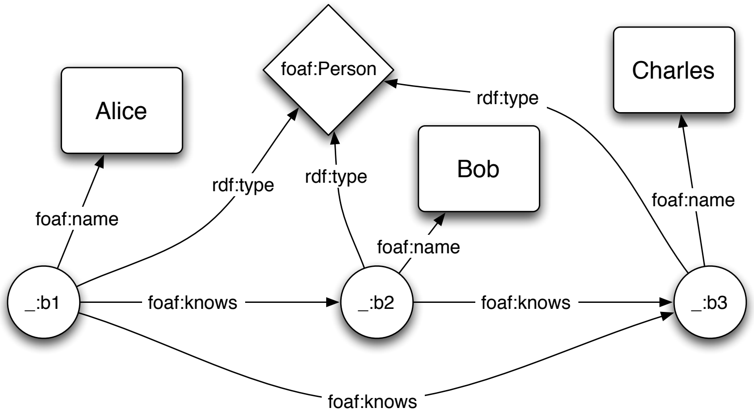
```
<relations>
  <person name="Alice">
    <knows>Bob</knows>
    <knows>Charles</knows>
  </person>
  <person name="Bob">
    <knows>Charles</knows>
  </person>
  <person name="Charles"/>
</relations>
```

relations.rdf

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:b1 a foaf:Person;
     foaf:name "Alice";
     foaf:knows _:b2;
     foaf:knows _:b3.
_:b2 a foaf:Person; foaf:name "Bob";
     foaf:knows _:b3.
_:b3 a foaf:Person; foaf:name "Charles".
```

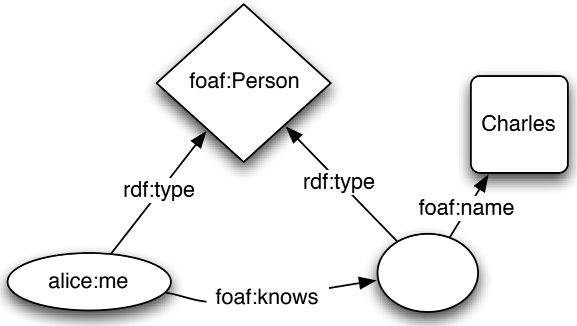
Lifting





So, why is XSLT, XQuery not enough?

- Because RDF ≠ RDF/XML !!!



1) many different RDF/XML representations…

```
<rdf:RDF xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <foaf:Person rdf:about="alice/me">
      <foaf:knows>
         <foaf:Person foaf:name="Charles"/>
      </foaf:knows>
   </foaf:Person>
</rdf:RDF>
```

```
<rdf:RDF xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description rdf:nodeID="x">
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
      <foaf:name>Charles</foaf:name>
   </rdf:Description>
   <rdf:Description rdf:about="alice/me">
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
      <foaf:knows rdf:nodeID="x"/>
   </rdf:Description>
</rdf:RDF>
```

```
<rdf:RDF xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description rdf:about="alice/me">
      <foaf:knows rdf:nodeID="x"/>
   </rdf:Description>
   <rdf:Description rdf:about="alice/me">
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="x">
      <foaf:name>Charles</foaf:name>
   </rdf:Description>
   <rdf:Description rdf:nodeID="x">
      <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   </rdf:Description>
</rdf:RDF>
```

2) … and actually a lot of RDF data residing in RDF stores, accessible via SPARQL endpoints already, rather than in RDF/XML

Our idea:

- New query language… but don't reinvent!

Xquery + SPARQL = **XSPARQL**

Prolog:	**P**	declare namespace *prefix*="*namespace-URI*" or prefix *prefix*: <*namespace-URI*>	
Body:	**F** **L** **W** **O**	for *var* in *XPath-expression* let *var* := *XPath-expression* where *XPath-expression* order by *expression*	**or**
	F' **D** **W** **M**	for *varlist* from / from named <*dataset-URI*> where {*pattern* } order by *expression* limit *integer* > 0 offset *integer* > 0	
Head:	**C**	construct { *template (with nested XSPARQL)* }	**or**
	R	return *XML + nested XSPARQL*	

Example: Mapping from RDF to XML

```
<relations>
{ for $Person $Name
  from <relations.rdf>
  where { $Person foaf:name $Name }
  order by $Name
  return
   <person name="{$Name}">
    {for $FName
     from <relations.rdf>
     where {
        $Person foaf:knows $Friend .
        $Person foaf:name $Name .
        $Friend foaf:name $Fname }
     return <knows>{$FName}</knows>
      } </person>
}</relations>
```

```
<relations>
  <person name="Alice">
    <knows>Bob</knows>
    <knows>Charles</knows>
  </person>
  <person name="Bob">
    <knows>Charles</knows>
  </person>
  <person name="Charles"/>
</relations>
```

Implementation and semantics:

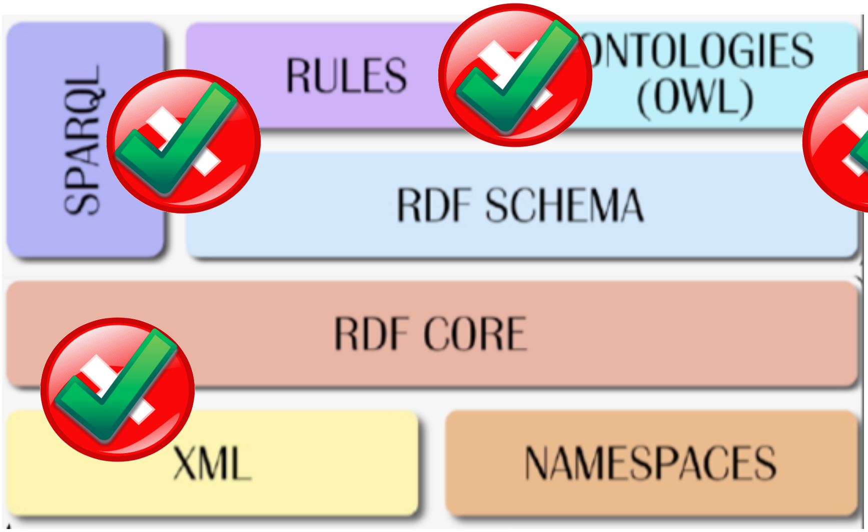
- Formal Semantics (XSPARQL1.0):
  - Based on XQuery formal Semantics
  - Can be implemented based on rewriting to XQuery

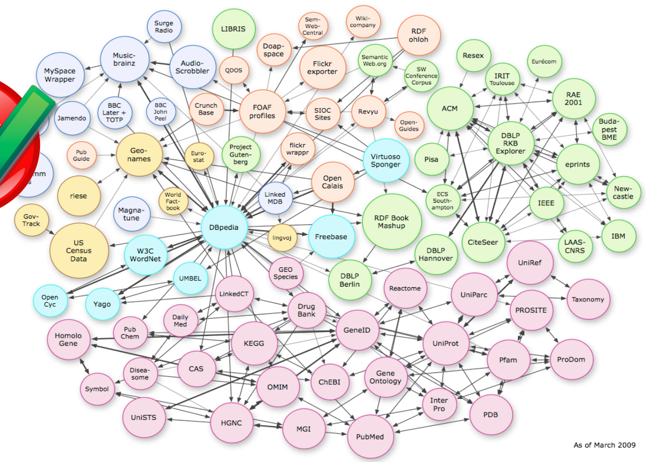
- Challenges/Limitations:
  - Nesting, scope of RDF dataset…
  - different "type systems" of RDF/XML (sequences)
  - adding ontological inference (to resolve heterogeneities)
  - We are working on this in XSPARQL1.1!

[ESWC2008], W3C member submission, Dipl.thesis [Bischof 2010]

How do the standards interplay?

- Challenges:
  - **1) Combining Rules & Ontologies**
  - **2) Querying Rules & Ontologies**
  - **3) Data on the Web is NOT clean/consistent!**
  - **4) Querying XML & RDF**





Take home message:

- *Deductive Database techniques/An Answer Set Programming based framework can cover large, useful parts of Semantic Web standards!*

- *Optimising/Integrating these with standard tools and languages (XQuery) and large scale data processing can be the basis for a Semantic Web middleware!*

Outlook:

- Hybrid KBs
  - Reasoning procedures for QEL, tractable fragments
  - Uniform equivalence definitions?
- SPARQL to Datalog
  - Potential for Optimisations?
  - Multi-set semantics?
  - SPARQL1.1 and beyond
- SAOR
  - SAOR vs. document-centric inference [Delbru et al. 2008]
- XSPARQL
  - Declarative underlying model, core fragment, integrating RDB, updates

Backup slides

Quantified Equilibrium Logics

- Based on first-order version of Logics of here- and- there (**QHT**), in the sense of a special two-world Kripke semantics for intuitionistic logic [van Dalen, 1983]

- QHT interpretations consist of pairs of first-order interpretations over the same domain, that both interpret constants the same.

- Slightly simplified, an **interpretation** $\mathcal{M} = \langle (D, \sigma), I_h, I_t \rangle$, s.t. $I_h \subseteq I_t$.
can be viewed as pair of **sets of atoms ⟨H, T⟩, such that H ⊆ T**

- The models are extended to all formulas via rules known in **intuitionistic logic**, logical consequence relation are the ones for (intuitionistic) Kripke semantics

- Quantified **Equilibrium** models are then those QHT models ⟨H, T⟩ where
  - H=T (**total**) and
  - there exists no QHT ⟨H', T⟩ model such that H' ⊊ H (**minimality**).

- Quantified Equilibrium Logics (**QEL**) is then QHT restricted to equilibrium models.

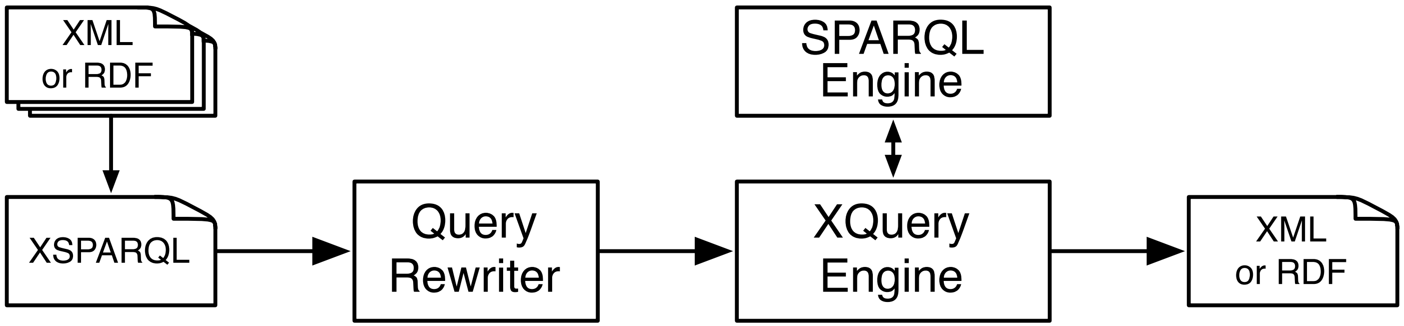
Quantified Here-and-there (QHT) Models:

For $w \in \{h, t\}$:

- $\mathcal{M}, w \models \varphi \wedge \psi$ iff $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$.
- $\mathcal{M}, w \models \varphi \vee \psi$ iff $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$.
- $\mathcal{M}, t \models \varphi \rightarrow \psi$ iff $\mathcal{M}, t \not\models \varphi$ or $\mathcal{M}, t \models \psi$.
- $\mathcal{M}, h \models \varphi \rightarrow \psi$ iff $\mathcal{M}, t \models \varphi \rightarrow \psi$ and $\mathcal{M}, h \not\models \varphi$ or $\mathcal{M}, h \models \psi$.
- $\mathcal{M}, w \models \neg\varphi$ iff $\mathcal{M}, t \not\models \varphi$.
- $\mathcal{M}, t \models \forall x \varphi(x)$ iff $\mathcal{M}, t \models \varphi(d)$ for all $d \in D$.
- $\mathcal{M}, h \models \forall x \varphi(x)$ iff $\mathcal{M}, t \models \forall x \varphi(x)$ and $\mathcal{M}, h \models \varphi(d)$ for all $d \in D$.
- $\mathcal{M}, w \models \exists x \varphi(x)$ iff $\mathcal{M}, w \models \varphi(d)$ for some $d \in D$.

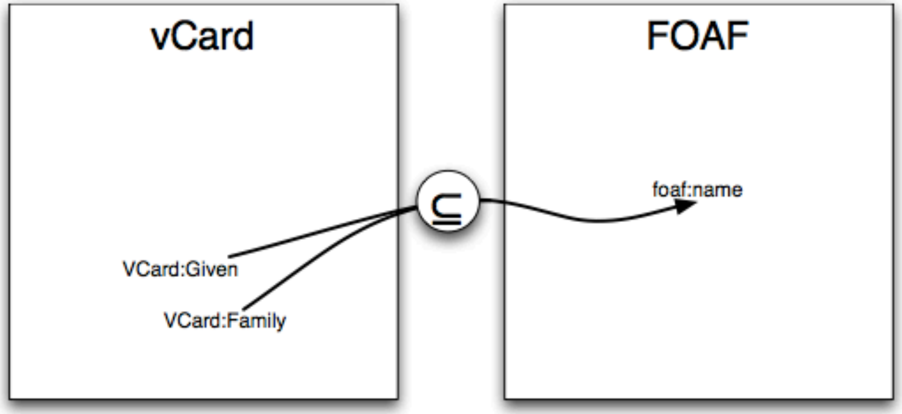
XSPARQL Implementation:

- Initial idea (and formalised in XSPARQL1.0):
  - extension of the XQuery semantics by plugging in SPARQL semantics in a modular way



  - Rewriting algorithm is defined for embedding XSPARQL into native XQuery plus interleaved calls to a SPARQL endpoint

SPARQL as a Rule Language for Ontology mappings **[ODBASE2007]**



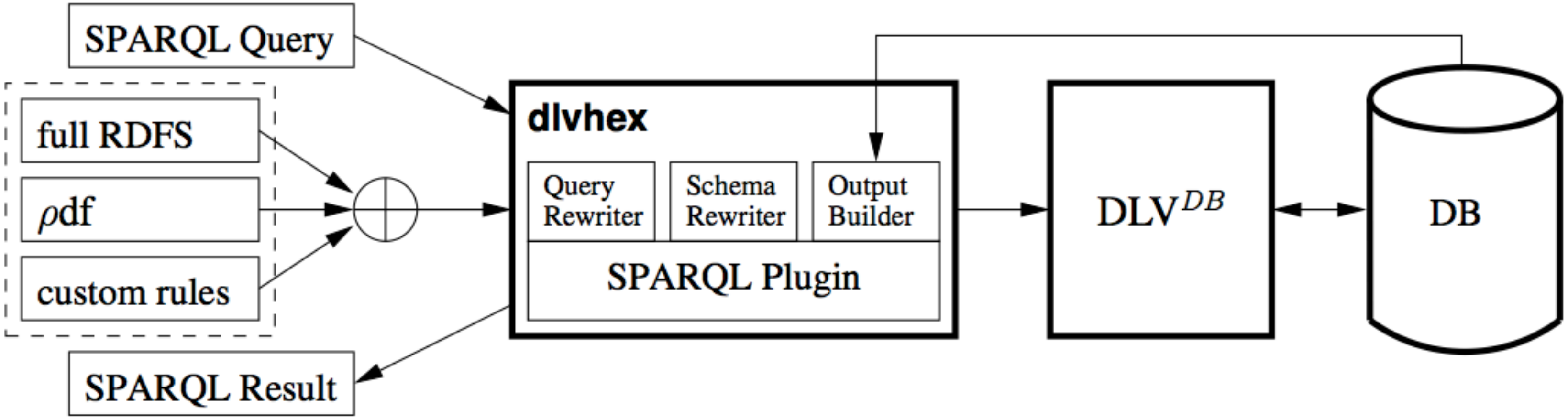

```
CONSTRUCT { ?X foaf:name fn:concat(?N," ",?F) }
WHERE     { ?X VCard:Given ?N. ?X VCard:Family ?F
          }
```

You rather want built-in functions in CONSTRUCT to describe ont. Mappings
But: Built-ins not expressible in SPARQL.
This is what SPARQL++ **[ODBASE2007]** provides, again by translating into Datalog^{not} with builtins.

Attention: Value generation in the CONSTRUCT part might again raise non-termination issues!

How to implement this? [ISWC2009]

- GiaBATA system:
  - SPARQL → dlvhex (logic program)
  - Additional inference rules (RDFS, etc.) → dlvhex (logic program) → SQL



- Deductive Database techniques:
  - Datalog engine (dlvhex)
  - Postgres SQL Database underneath (dlv-db)
  - RDF storable in different schemas in RDB
  - Magic sets, storage

XSPARQL Formal Semantics

- Current formalisation embeds rewriting in the functional semantics of XQuery: http://xsparql.deri.org/spec/xsparql-semantics.html#id:flwor-expressions

mapping rules $[\cdot]_{Expr'}$ inherit from the definitions of XQuery's $[\cdot]_{Expr}$

$$\left[\begin{array}{l}\texttt{for}\ \$VarName_1\ \ldots\ \$VarName_n\ DatasetClause\ \texttt{where}\\ GroupGraphPattern\ SolutionModifier\ ReturnClause\end{array}\right]_{Expr'}$$

$$==$$

$$\left\ulcorner\begin{array}{l}\texttt{let}\ \$\_aux\_queryresult\ \texttt{:=}\\ \left[\begin{array}{l}VarName_1\ \ldots\ VarName_n\ DatasetClause\\ \texttt{where}\ GroupGraphPattern\ SolutionModifier\end{array}\right]_{SparqlQuery}\end{array}\right\urcorner$$

$$\left[\begin{array}{l}\$VarName_1\ \ldots\ \$VarName_n\ DatasetClause\\ \texttt{where}\ GroupGraphPattern\ SolutionModifier\end{array}\right]_{SparqlQuery}$$

$$==$$

$$fs{:}sparql\left(\left[\begin{array}{l}\texttt{fs:serialize("SELECT}\ \$VarName_1\ \ldots\ \$VarName_n\texttt{"},\ DatasetClause,\texttt{" where \{ "},\\ \quad\texttt{fs:serialize(}\ GroupGraphPattern\ \texttt{), " \}}\ SolutionModifier\texttt{")}\end{array}\right]_{Expr'}\right)$$

Rewriting XSPARQL to XQuery…

```
construct { _:b foaf:name {fn:concat("""",$N," ",$F,"""")} }
from <vcard.rdf>
where { $P vc:Given $N . $P vc:Family $F . }
```

```
let $aux_query := fn:concat("http://localhost:2020/sparql?query=",
                  fn:encode-for-uri(
                   "select $P $N $F from <vcard.rdf>
                    where {$P vc:Given $N. $P vc:Family $F.}"))
for $aux_resu
        in doc($aux_query)//sparql_result:result
let $P_Node :                          :binding[@name="P"]
let $N_Node :                          :binding[@name="N"]
let $F_Node := $aux_result/sparql_result:binding[@name="F"]
let $N := data($N_Node/*)let $N_NodeType := name($N_Node/*)
let $N_RDFTerm := local:rdf_term($N_NodeType,$N)
...
return ( fn:c                                        "),
         ( f                                         """") ), "." )
```

1. Encode SPARQL in HTTP call SELECT Query

2. Execute call, via fn:doc function

3. Collect results from SPARQL result format (XML)

4. `construct` becomes `return` that outputs triples.

Submission acknowledged by W3C

XSPARQL Language Specification

W3C Member Submission

XSPARQL Language Specification

W3C Member Submission 20 January 2009

This version:
http://www.w3.org/Submission/2009/SUBM-xsparql-spec-20090120/
Latest version:
http://www.w3.org/Submission/xsparql-spec/
Authors:
Axel Polleres - DERI, NUI Galway
Thomas Krennwallner - Institute of Information Systems, Vienna University of Technology
Nuno Lopes - DERI, NUI Galway
Jacek Kopecký - University of Innsbruck
Stefan Decker - DERI, NUI Galway

This work is supported by Science Foundation Ireland under grants number SFI/02/CE1/I131 and SFI/08/CE/I1380 and under the European Commission European FP6 project inContext (IST-034718).

W3C XHTML + RDFa

Abstract

XSPARQL is a query language combining XQuery and SPARQL for transformations between RDF and XML. This document contains provides a language description of XSPARQL.

- 4 Documents:
  - Language Spec.
  - Semantics
  - Implementation& Test-Cases
  - Use Cases
- Online prototype
  http://xsparql.deri.org
- Supporters
  - Asemantics
  - INRIA
  - OpenLink
  - Ontotext
  - CTIC
  - Profium
  - Talis
  - Univ. Innsbruck

2. RDF can be described in terms of Ontologies and Rules ➔ *allows Reasoning!*

OWL's theoretical foundation: Description Logics,

SHOIN [Horrocks and Patel-Schneider, 2004]

SROIQ [Horrocks et al. 2006]

RIF's theoretical foundation: Logic programming, F-Logic,

but also Datalog/Answer Set Programming, Deductive Databases

(some RIF dialects allow negation as failure)

RDF Schema: in essence in the intersection

(but strictly speaking more liberal than Description Logics)

RDF is the basis for Linked Data:

1. Everything gets a URI (conferences, people, talks, …)
2. These URIs are linked via RDF describing relations
3. Relations are URIs again (e.g. :name)
4. When I dereference the URIs, I should find more information about them