The Role of Logics and Logic Programming in Semantic Web Standards (OWL2, RIF, SPARQL1.1)

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

OÉ Gaillimh NUI Galway

science foundation ireland
fondúireacht eolaíochta éireann

Outline

- Quick intro of RDF/OWL/Linked Open Data
- OWL2
  - Overview: from OWL 1 to OWL 2
  - Reasoning services in OWL 2
  - OWL2 Treactable Fragments (OWL2RL, OWL2EL, OWL2QL)
- OWL2 and RIF
- OWL2 and SPARQL1.1

- Time allowed: Implementing SPARQL, OWL2RL, RIF on top of DLV – The GiaBATA system

Example: Finding experts/reviewers?

***Tim Berners-Lee**, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269*

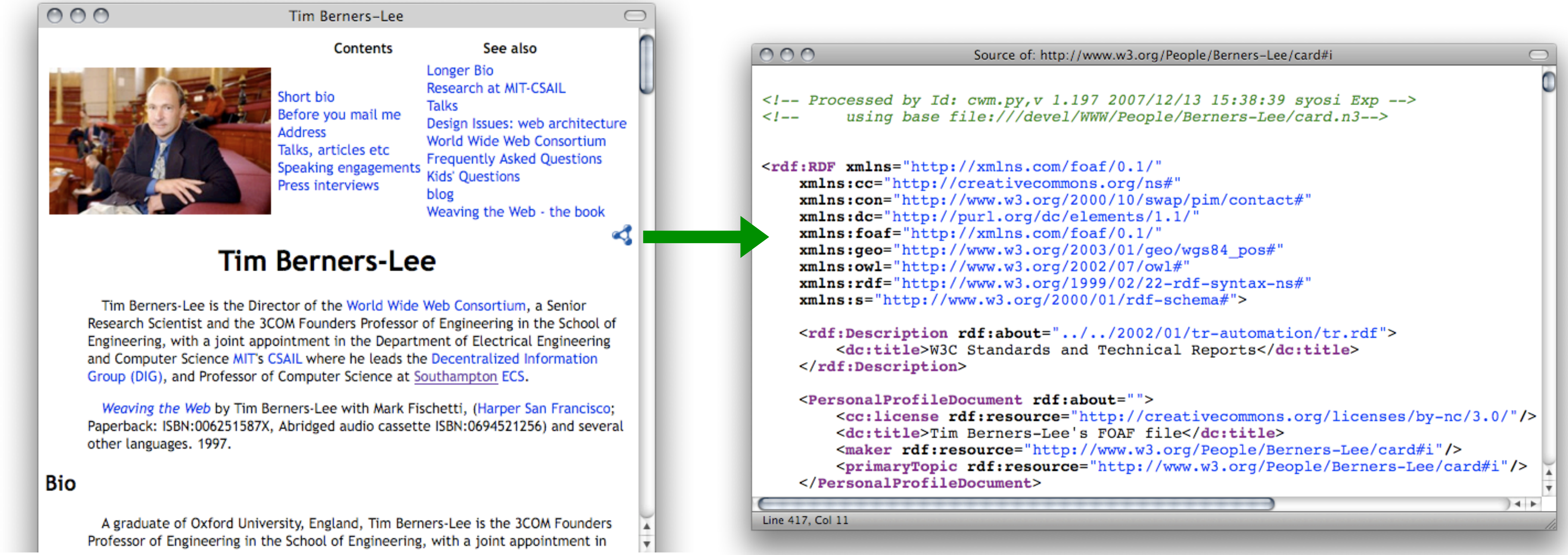
- Who are the right reviewers? Who has the right expertise?
- Which reviewers are in conflict?
- Observation: Most of the necessary data already on the Web, as **RDF**!

- More and more of it follows the **Linked Data principles**, i.e.:
  1. Use URIs as names for things
  2. Use HTTP dereferenceable URIs so that people can look up those names.
  3. When someone looks up a URI, provide useful information.
  4. Include links to other URIs so that they can discover more things.

RDF on the Web

- **(i) directly by the publishers**
- (ii) by e.g. transformations, D2R, RDFa exporters, etc.

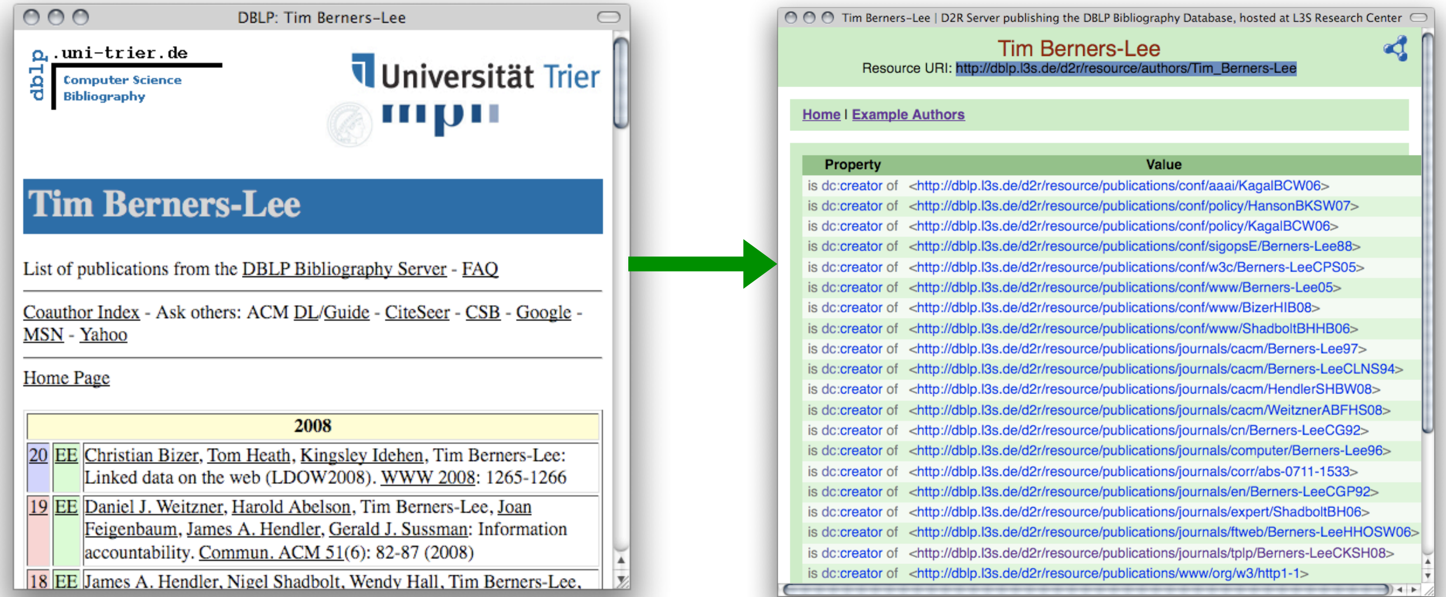
FOAF/RDF linked from a home page: personal data (foaf:name, foaf:phone, etc.), relationships foaf:knows, rdfs:seeAlso)



RDF on the Web

- (i) directly by the publishers
- **(ii) by e.g. transformations, D2R, RDFa exporters, etc.**

e.g. L3S' RDF export of the DBLP citation index, using FUB's D2R (http://dblp.l3s.de/d2r/)



Gives unique URIs to authors, documents, etc. on DBLP! E.g.,

http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee,

http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08

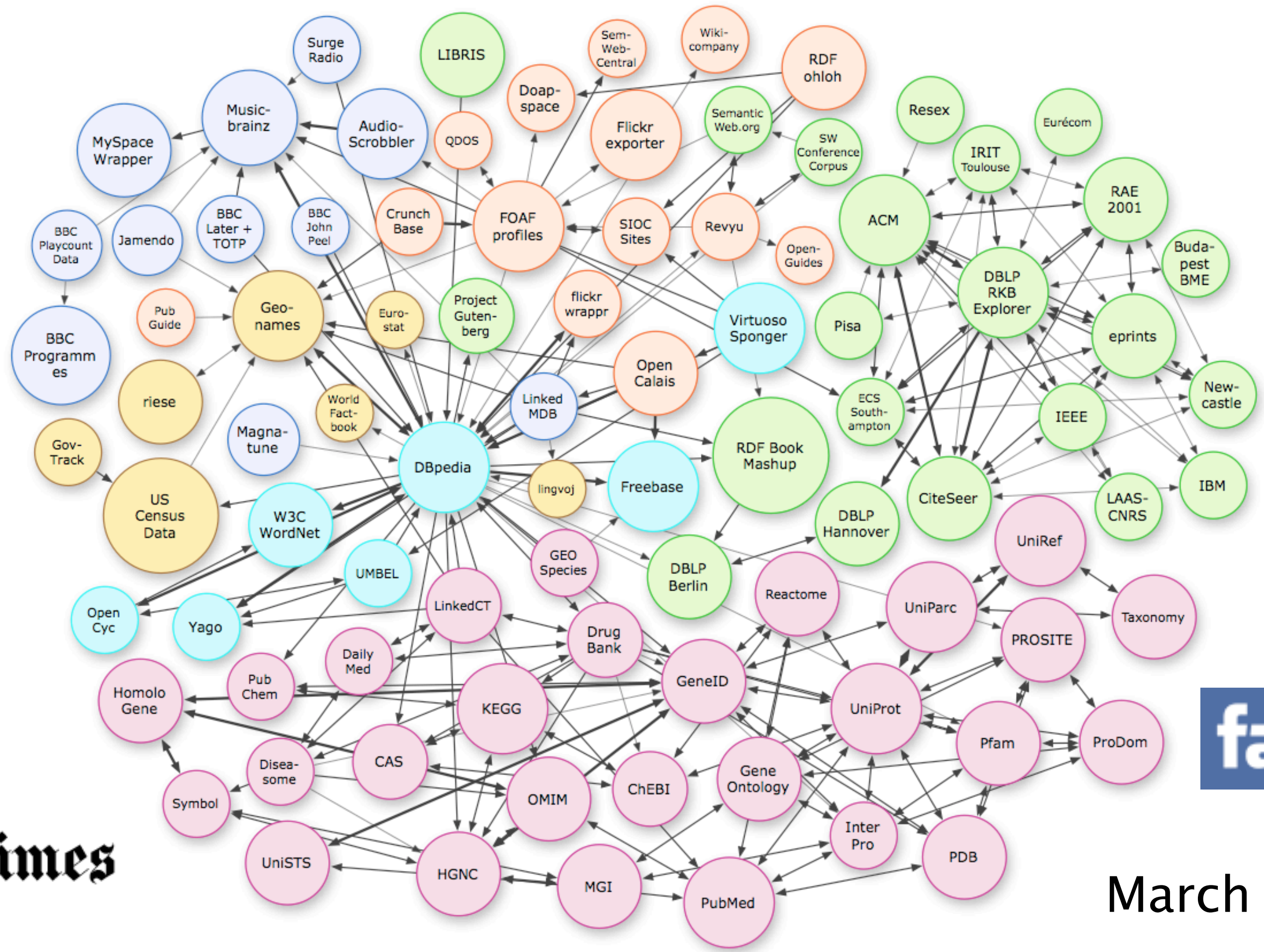
Provides RDF version of all DBLP data + query interface!

Linked Open Data

Miscellaneous YAHOO! Google

dblp.uni-trier.de Computer Science Bibliography

WIKIPEDIA



Freebase

. . .

The New York Times

facebook

March 2009

- Excellent tutorial here: http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

OÉ Gaillimh NUI Galway

sfi science foundation ireland fondúireacht eolaíochta éireann

DERI

RDF Data online: Example

- Data in RDF: **Triples**

subject predicate object

  - DBLP:

```
<http://dblp.l3s.de/…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.
<http://dblp.l3s.de/…/journals/tplp/Berners-LeeCKSH08> dc:creator
     <http://dblp.l3s.de/d2r/…/Tim_Berners-Lee> .
  …
<http://dblp.l3s.de/d2r/…/Tim_Berners-Lee> foaf:homepage
       <http://www.w3.org/People/Berners-Lee/> .

…
<http://dblp.l3s.de/d2r/…/Dan_Brickley> foaf:name "Dan Brickley"^^xsd:string.
```

  - Tim Berners-Lee's FOAF file:

```
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows
    <http://dblp.l3s.de/d2r/…/Dan_Brickley> .
<http://www.w3.org/People/Berners-Lee/card#i> rdf:type foaf:Person .
<http://www.w3.org/People/Berners-Lee/card#i> foaf:homepage
    <http://www.w3.org/People/Berners-Lee/> .
```

How can I query such data? SPARQL

- SPARQL – W3C approved standardized query language for RDF:
  - look-and-feel of “SQL for the Web”
  - allows to ask queries like
    - “All documents by Tim Berners-Lee”

      …

Example:

```
SELECT ?D
FROM <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>
WHERE {?D dc:creator <http://dblp.l3s.de/…/authors/Tim_Berners-Lee>}
```

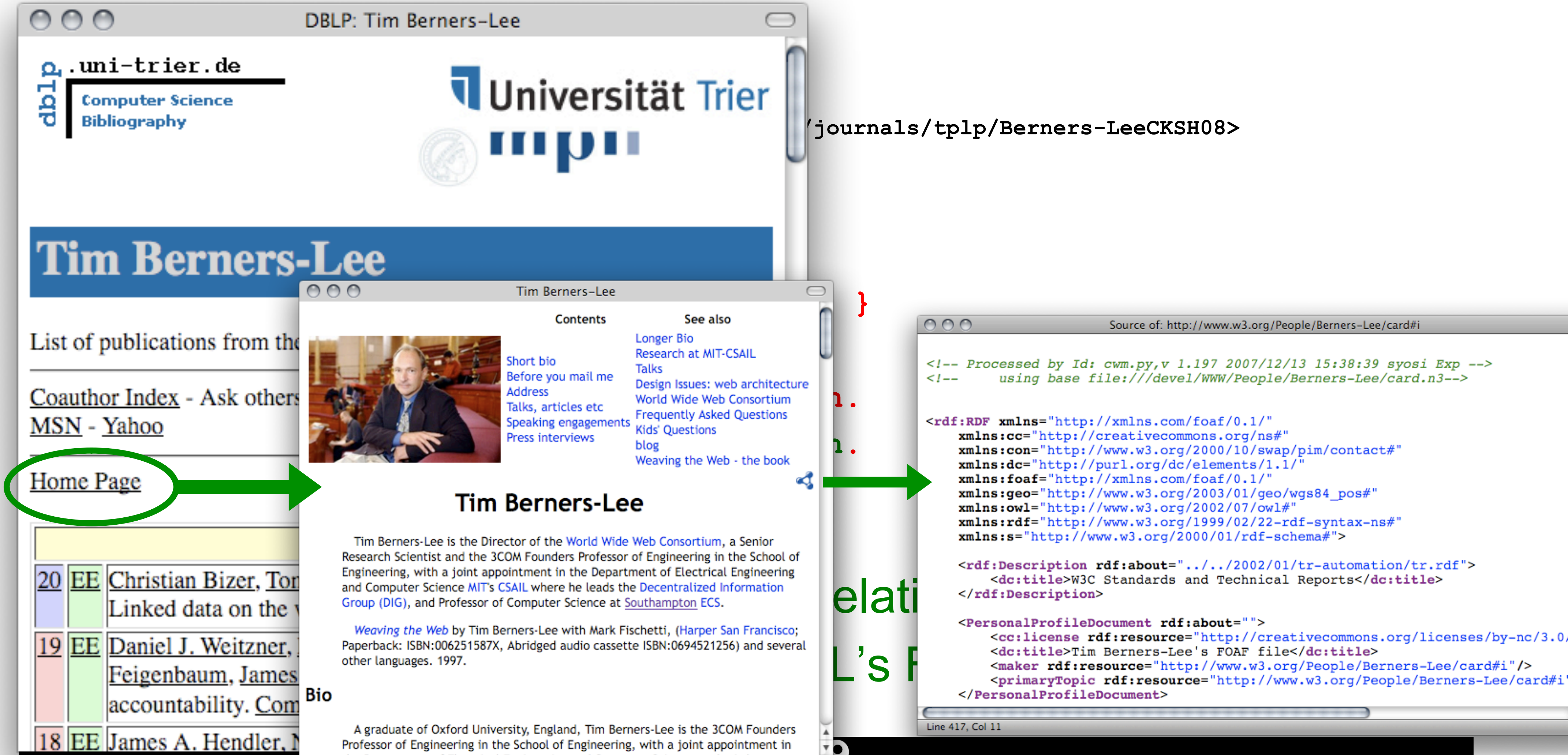
SPARQL more complex patters: e.g. CQs

- "**Names** ***of all persons who co-authored with authors of http:// dblp.l3s.de/d2r/…/Berners-LeeCKSH08***"

```
SELECT ?Name WHERE
   { <http://dblp.l3s.de/d2r/resource/publication/journals/tplp/Berners-LeeCKSH08>
        dc:creator ?Author.
        ?D dc:creator ?Author.
        ?D dc:creator ?CoAuthor.
        ?CoAuthor foaf:name ?Name
   }
```

SPARQL more complex patters: e.g. UCQs

- "Names *of all persons who* **co-authored** *with authors of http://dblp.l3s.de/d2r/…/Berners-LeeCKSH08* **or known by co-authors**"



Back to the Data:

- DBLP:

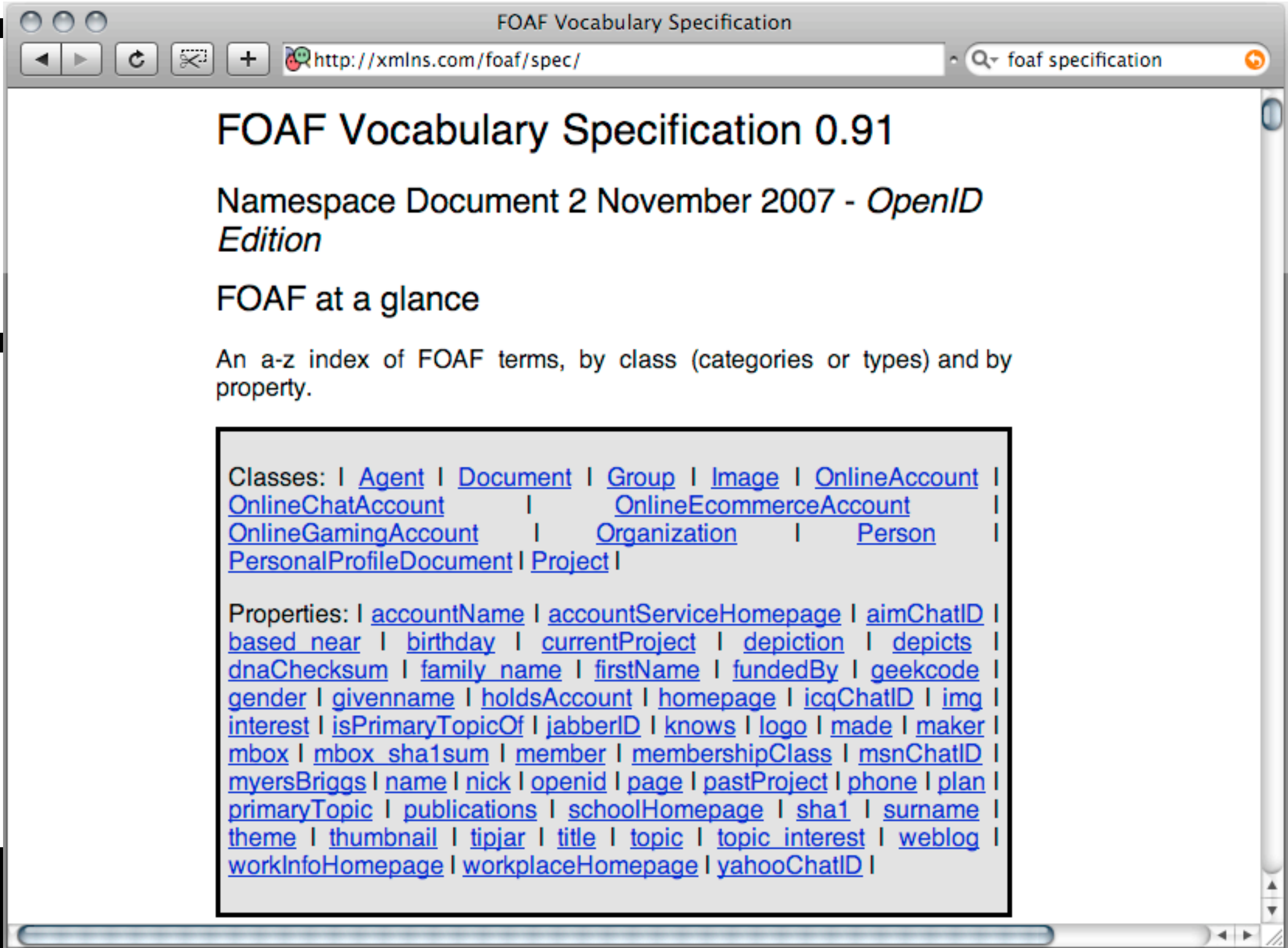
```
<http://dblp.l3s.de/…/journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.
<http://dblp.l3s.de/…/journals/tplp/Berners-LeeCKSH08> dc:creator
     <http://dblp.l3s.de/d2r/…/Tim_Berners-Lee> .
  …
<http://dblp.l3s.de/d2r/…/Tim_Berners-Lee> foaf:homepage
       <http://www.w3.org/People/Berners-Lee/> .
```

- Tim Berners-Lee's FOAF file:

```
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows
    <http://dblp.l3s.de/d2r/…/Dan_Brickley> .
<http://www.w3.org/People/Berners-Lee/card#i> foaf:homepage
    <http://www.w3.org/People/Berners-Lee/> .
```

- Even if I have the FOAF data, I cannot answer the query:
  - Different identifiers used for Tim Berners-Lee
  - Who tells me that Dan Brickley is a foaf:Person?
- Linked Data needs **Reasoning**!

Reasoning on Semantic Web Data



FOAF Vocabulary Specification 0.91

Namespace Document 2 November 2007 - *OpenID Edition*

FOAF at a glance

An a-z index of FOAF terms, by class (categories or types) and by property.

Classes: | Agent | Document | Group | Image | OnlineAccount | OnlineChatAccount | OnlineEcommerceAccount | OnlineGamingAccount | Organization | Person | PersonalProfileDocument | Project |

Properties: | accountName | accountServiceHomepage | aimChatID | based_near | birthday | currentProject | depiction | depicts | dnaChecksum | family_name | firstName | fundedBy | geekcode | gender | givenname | holdsAccount | homepage | icqChatID | img | interest | isPrimaryTopicOf | jabberID | knows | logo | made | maker | mbox | mbox_sha1sum | member | membershipClass | msnChatID | myersBriggs | name | nick | openid | page | pastProject | phone | plan | primaryTopic | publications | schoolHomepage | sha1 | surname | theme | thumbnail | tipjar | title | topic | topic_interest | weblog | workInfoHomepage | workplaceHomepage | yahooChatID |

Ontologies: Example FOAF

foaf:knows rdfs:domain foaf:Person

$$\exists knows.\top \sqsubseteq Person$$

foaf:knows rdfs:range foaf:Person

$$\exists knows^{-}.\top \sqsubseteq Person$$

foaf:Person rdfs:subclassOf foaf:Agent

$$Person \sqsubseteq Agent$$

foaf:homepage rdf:type owl:inverseFunctionalProperty .

$$\top \sqsubseteq \leq 1 homepage^{-}$$

...

RDFS inference by rules 1/2

- Semantics of RDFS can be partially expressed as (Datalog like) rules:

```
rdfs1: triple(S,rdf:type,C) :- triple(S, P, O), triple(P, rdfs:domain, C)
rdfs2: triple(O,rdf:type,C) :- triple(S, P, O), triple(P, rdfs:range, C)
```

```
rdfs3: triple(S,rdf:type,C2):-triple(S,rdf:type,C1),triple(C1,rdfs:subclassOf,C2)
```

cf. informative Entailment rules in [RDF-Semantics, W3C, 2004], [Muñoz et al. 2007]

RDFS inference by rules 1/2

- Semantics of RDFS can be partially expressed as (Datalog like) rules:

```
rdfs1: { ?S rdf:type ?C } :- { ?S ?P ?O . ?P rdfs:domain ?C . }
rdfs2: { ?O rdf:type ?C } :- { ?S ?P ?O . ?P rdfs:range ?C . }
```

```
rdfs3: { ?S rdf:type ?C2 } :- {?S rdf:type ?C1 . ?C1 rdfs:subclassOf ?C2 . }
```

cf. informative Entailment rules in [RDF-Semantics, W3C, 2004], [Muñoz et al. 2007]

RDFS+OWL inference by rules 2/2

- OWL Reasoning e.g. inverseFunctionalProperty can also (partially) be expressed by Rules:

```
owl1: { ?S1 owl:SameAs ?S2 } :-
          { ?S1 ?P ?O . ?S2 ?P ?O . ?P rdf:type owl:InverseFunctionalProperty }


owl2: { ?Y ?P ?O } :- { ?X owl:SameAs ?Y . ?X ?P ?O }
owl3: { ?S ?Y ?O } :- { ?X owl:SameAs ?Y . ?S ?X ?O }
owl4: { ?S ?P ?Y } :- { ?X owl:SameAs ?Y . ?S ?P ?X }
```

cf. pD* fragment of OWL, [ter Horst, 2005], SAOR [Hogan et al. 2009] or, more recent: **OWL2 RL**

RDFS+OWL inference by rules: Example:

- By rules of the previous slides we can infer additional information needed, e.g.

TimBL's FOAF: `<…/Berners-Lee/card#i> foaf:knows <…/Dan_Brickley> .`
FOAF Ontology: `foaf:knows rdfs:range foaf:Person`

by rdfs2 → `<…/Dan_Brickley> rdf:type foaf:Person.`

TimBL's FOAF: `<…/Berners-Lee/card#i> foaf:homepage <http://www.w3.org/People/Berners-Lee/> .`
DBLP: `<…/dblp.l3s.de/d2r/…/Tim_Berners-Lee> foaf:homepage <http://www.w3.org/People/Berners-Lee/> .`
FOAF Ontology: `foaf:homepage rdfs:type owl:InverseFunctionalProperty.`

by owl1 → `<…/Berners-Lee/card#i> owl:sameAs <…/Tim_Berners-Lee>.`

- Who tells me that Dan Brickley is a foaf:Person? → solved!
- Different identifiers used for Tim Berners-Lee → solved!

RDFS+OWL inference, what's missing?

- Note: Not all of OWL Reasoning can be expressed in Datalog straightforwardly, e.g.:

```
foaf:Person owl:disjointWith foaf:Organisation
```

Can be written/and reasoned about with FOL/DL reasoners:

FOL Syntax: $\forall X.Person(X) \supset \neg Organisation(X)$

DL Syntax: $Person \sqcap Organisation \sqsubseteq \bot$

Problem: Inconsistencies! **Complete** FOL/DL reasoning is often not suitable per se for Web data… [Hogan et al.2009,]

But can be "approximated" by Rules (without explosion):

```
owl5: ERROR :- { ?X a ?C1; a ?C2. ?C1 owl:disjointWith ?C2.}
```

The more "common" view on OWL…

Expressing property characteristics:

OWL property axioms as RDF triples	DL syntax	FOL short representation
P rdfs:domain C .	$\top \sqsubseteq \forall P^{-}.C$	$\forall x, y. P(x, y) \supset C(x)$
P rdfs:range C .	$\top \sqsubseteq \forall P.C$	$\forall x, y. P(x, y) \supset C(y)$
P owl:inverseOf P_0 .	$P \equiv P_0^{-}$	$\forall x, y. P(x, y) \equiv P_0(y, x)$
P rdf:type owl:SymmetricProperty.	$P \equiv P^{-}$	$\forall x, y. P(x, y) \equiv P(y, x)$
P rdf:type owl:FunctionalProperty.	$\top \sqsubseteq \leqslant 1P$	$\forall x, y, z. P(x, y) \wedge P(x, z) \supset y = z$
P rdf:type owl:InverseFunctionalProperty.	$\top \sqsubseteq \leqslant 1P^{-}$	$\forall x, y, z. P(x, y) \wedge P(z, y) \supset x = z$
P rdf:type owl:TransitiveProperty.	$P^{+} \sqsubseteq P$	$\forall x, y, z. P(x, y) \wedge P(y, z) \supset P(x, z)$

Expressing complex class descriptions:

OWL complex class descriptions*	DL syntax	FOL short representation
owl:Thing	$\top$	$x = x$
owl:Nothing	$\bot$	$\neg x = x$
owl:intersectionOf $(C_1 \ldots C_n)$	$C_1 \sqcap \cdots \sqcap C_n$	$C_1(x) \wedge \cdots \wedge C_n(x)$
owl:unionOf $(C_1 \ldots C_n)$	$C_1 \sqcup \cdots \sqcup C_n$	$C_1(x) \vee \cdots \vee C_n(x)$
owl:complementOf (C)	$\neg C$	$\neg C(x)$
owl:oneOf $(o_1 \ldots o_n)$	$\{o_1, \ldots, o_n\}$	$x = o_1 \vee \cdots \vee x = o_n$
owl:restriction (P owl:someValuesFrom (C))	$\exists P.C$	$\exists y. P(x, y) \wedge C(y)$
owl:restriction (P owl:allValuesFrom (C))	$\forall P.C$	$\forall y. P(x, y) \supset C(y)$
owl:restriction (P owl:value (o))	$\exists P.\{o\}$	$P(x, o)$
owl:restriction (P owl:minCardinality (n))	$\geqslant nP$	$\exists y_1 \ldots y_n . \bigwedge_{k=1}^{n} P(x, y_k) \wedge \bigwedge_{i<j} y_j \neq y_j$
owl:restriction (P owl:maxCardinality (n))	$\leqslant nP$	$\forall y_1 \ldots y_{n+1} . \bigwedge_{k=1}^{n+1} P(x, y_k) \supset \bigvee_{i<j} y_i = y_j$

* For reasons of legibility, we use a variant of the OWL abstract syntax [Patel-Schneider *et al.*, 2004] in this table.

Why OWL1 is Not Enough

- Too expensive to reason with
  - High complexity: NEXPTIME-complete
  - The most lightweight sublanguage OWL-Lite is **NOT** lightweight
  - Some ontologies only use some limited expressive power; e.g. The SNOMED (Systematised Nomenclature of Medicine) ontology
- Not expressive enough; e.g.
  - No user defined datatypes [Pan 2004; Pan and Horrocks 2005; Motik and Horrocks 2008]
  - No metamodeling support [Pan 2004; Pan, Horrocks, Schreiber, 2005; Motik 2007]
  - Limited property support [Horrocks et al., 2006]

From OWL1 to OWL2

- OWL2: A new version of OWL
- Main goals:

1. To define "profiles" of OWL that are:
   - smaller, easier to implement and deploy
   - cover important application areas and are easily understandable to non-expert users

2. To add a few extensions to current OWL that are useful, and is known to be implementable
   - many things happened in research since 2004

OWL2

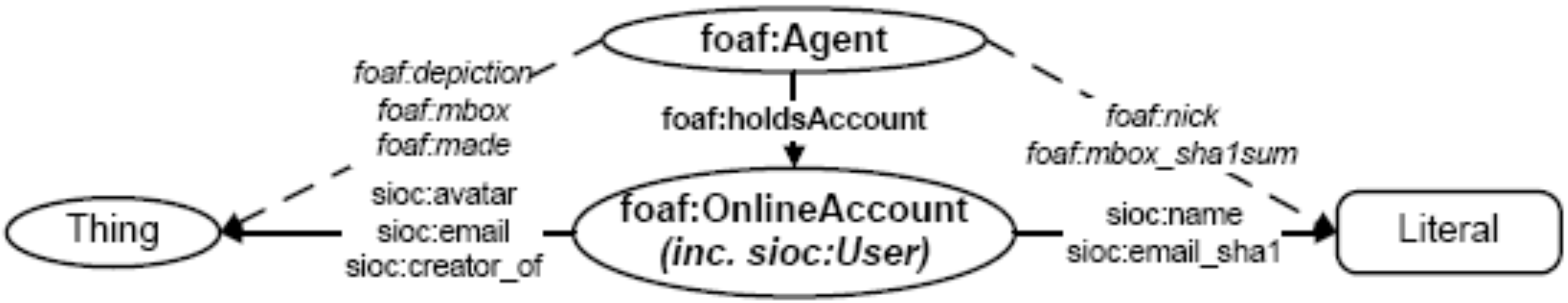
HOT! NEW!

Common ontologies on the Web don't use it a lot as of yet…

… but adds interesting functionality, potentially useful for Web ontologies, e.g.

- PropertyChains
  - E.g. could be useful to tie sioc:name and foaf:nick via foaf:holdsAccount:

```
foaf:nick owl:propertyChainAxiom (foaf:holdsAccount sioc:name)
```



OWL2

HOT!
NEW!

Common ontologies on the Web don't use it a lot as of yet…

… but adds interesting functionality, potentially useful for Web ontologies, e.g.

- hasKey:
  - *Multi-attribute Keys now possible in OWL, e.g.*
    - `foaf:OnlineAccount/sioc:User` members are uniquely identified by a combination of `foaf:accountName` and `foaf:accountServiceHomepage`:

```
foaf:OnlineAccount owl:hasKey
    (foaf:accountName foaf:accountServiceHomepage) .
```

New Expressiveness in OWL 2

- New expressive power
  - user defined datatypes, e.g.:

    "Ages are integers between 0 and 150"

```
personAge owl:equivalentClass _:x
_:x rdf:type rdfs:Datatype
_:x owl:onDatatype xsd:integer
_:x owl:withRestrictions (_:y1 _:y2)
_:y1 xsd:minInclusive "0"^^xsd:integer
_:y2 xsd:maxInclusive "150"^^xsd:integer
```

  - punning (metamodeling), e.g.:

```
John rdf:type Father
Father rdf:type SocialRole
```

New Expressiveness in OWL 2

- New expressive power on properties
  - qualified cardinality restrictions, e.g.:

```
_:x rdf:type owl:Restriction
_:x owl:onProperty foaf:knows
_:x owl:minQualifiedCardinality "10"^^xsd:nonNegativeInteger
_:x owl:onClass Irish
```

  - property chain inclusion axioms, e.g.:

```
foaf:nick owl:propertyChainAxiom (foaf:holdsAccount sioc:name)
```

  - local reflexivity restrictions, e.g.:

```
_:x rdf:type owl:Restriction
_:x owl:onProperty like
_:x owl:hasSelf "true"^^xsd:boolean
```

[for narcissists]

  - reflexive, irreflexive, symmetric, and antisymmetric properties, e.g.:

```
foaf:know rdf:type owl:ReflexiveProperty
rel:childOf rdf:type owl:IrreflexiveProperty
```

  - disjoint properties, e.g.:

```
rel:childOf owl:propertyDisjointWith rel:parentOf
```

  - keys, e.g.:

```
foaf:OnlineAccount owl:hasKey
(foaf:accountName foaf:accountServiceHomepage)
```

New Expressiveness in OWL 2

- Syntactic sugar (make things easier to say)
  - Disjoint unions, e.g.:

    ```
    child owl:disjointUnionOf (boy girl)
    ```

  - Disjoint classes, e.g.:

    ```
    _:x rdf:type owl:AllDisjointClasses
    _:x owl:members (boy girl)
    ```

  - Negative assertions, e.g.:

    ```
    _:x rdf:type owl:NegativePropertyAssertion
    _:x owl:sourceIndividual John
    _:x owl:assertionProperty foaf:know
    _:x owl:targetIndividual Mary
    ```

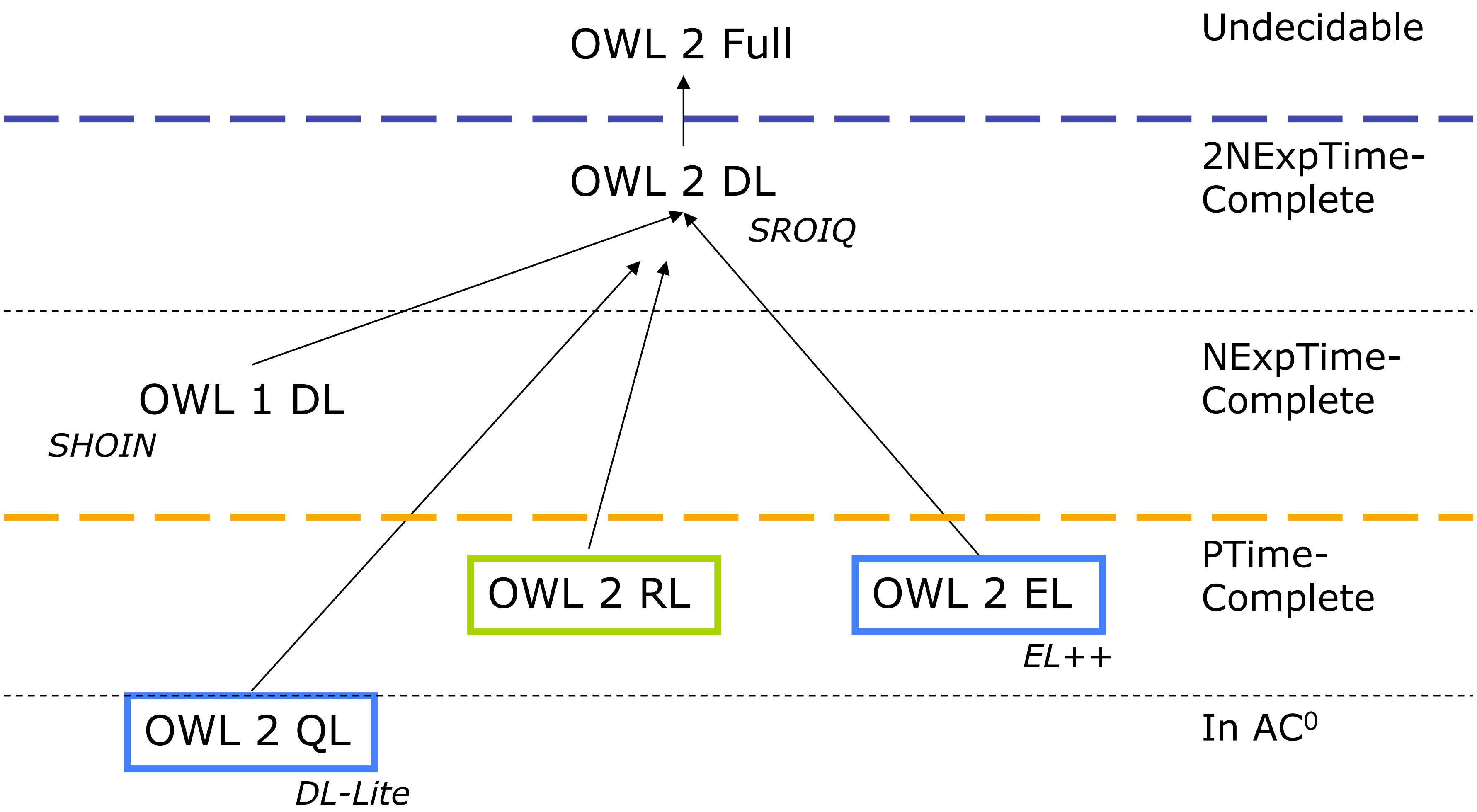
OWL 2 DL

- $\mathcal{R}$ often used for $\mathcal{ALC}$ extended with property chain inclusion axioms
  - following the notion introduced in $\mathcal{RIQ}$ [Horrocks and Sattler, 2003]
  - including transitive property axioms
- Additional letters indicate other extensions, e.g.:
  - $\mathcal{S}$ for property characteristics (e.g., reflexive and symmetric)
  - $\mathcal{O}$ for **nominals**/singleton classes
  - $\mathcal{I}$ for inverse roles
  - $\mathcal{Q}$ for qualified number restrictions
- property characteristics ($\mathcal{S}$) + $\mathcal{R}$ + nominals ($\mathcal{O}$) + inverse ($\mathcal{I}$) + qualified number restrictions($\mathcal{Q}$) = $\mathcal{SROIQ}$
- $\mathcal{SROIQ}$ [Horrocks et al., 2006] is the basis for OWL 2 DL
- Available Reasoners: Hermit (Oxford), Pellet (Clark&Parsia)

OWL 2 Profiles and Reasoning Services

- Rationale:
  - Tractable
  - Tailored to specific reasoning services
- Popular reasoning services
  - ABox reasoning: OWL 2 RL
  - TBox reasoning: OWL 2 EL
  - Query answering: OWL 2 QL
- Specification: http://www.w3.org/TR/2009/owl2-profiles/

The family tree



OWL2RL

- Maximal fragment of OWL expressible in Horn Rules
  - Rules for subclassing, subproperties, propChains, (inverse) functionalProperties, hasValue…
  - No support for arbitrary card restrictions, existentials in rule heads, etc.
  - See before… more later…
- See also discussion of the rule set in [Hogan&Decker, 2009]

OWL 2 EL

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime (**PTime-Complete**)
  - Data complexity of query answering also PTime-Complete
- Based on $\mathcal{EL}$ family of description logics [Baader et al. 2005]
- Can exploit **saturation** based reasoning techniques
  - Computes complete classification in “one pass”
  - Computationally optimal (PTime for EL)
  - Can be extended to Horn fragment of OWL DL [Kazakov 2009]

Saturation-based Technique (basics)

- Normalise ontology axioms to standard form:

  $A \sqsubseteq B \quad A \sqcap B \sqsubseteq C \quad A \sqsubseteq \exists R.B \quad \exists R.B \sqsubseteq C$

- Saturate using inference rules:

$$\frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C} \qquad \frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

- Extension to Horn fragment requires (many) more rules

Saturation-based Technique (basics)

Example:

$$\frac{A \sqsubseteq B \quad B \sqsubseteq C}{A \sqsubseteq C} \qquad \frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C \quad \exists R.C \sqsubseteq D}{A \sqsubseteq D}$$

```
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Agent rdfs:subClassOf owl:Thing.

foaf:Person rdfs:subclassOf
                 [ a owl:Restriction ;
          owl:onProperty :hasFather ;
          owl:someValuesFrom foaf:Person. ].

:hasFather domain :Child.
```

$$Person \sqsubseteq \top$$

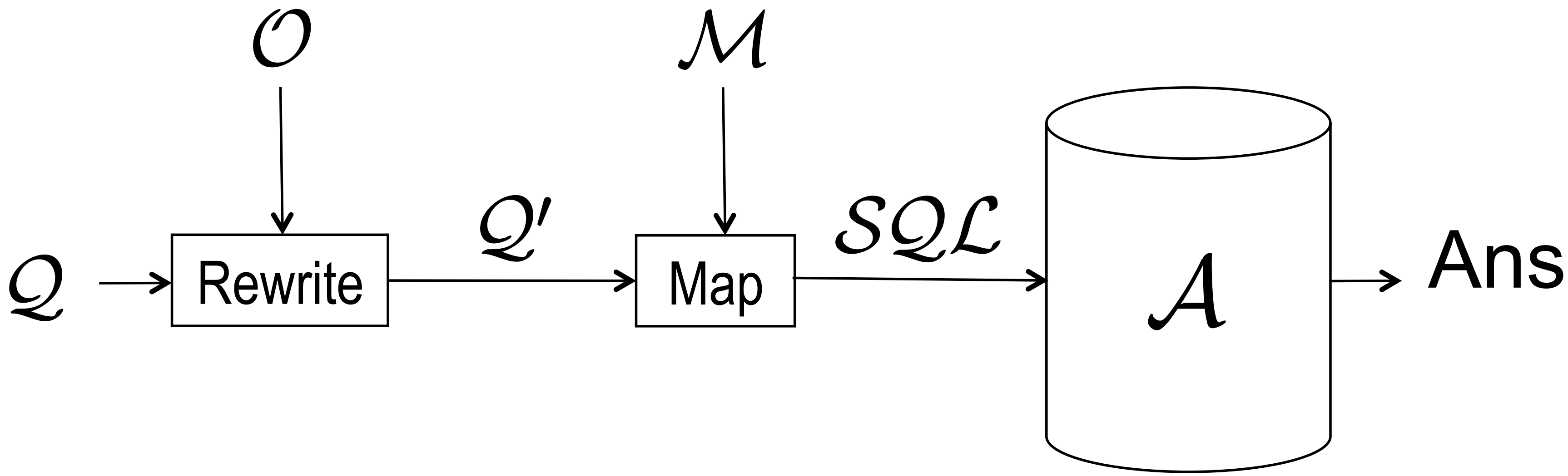
$$Person \sqsubseteq Child$$

OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in **AC^0**
- Based on **DL-Lite** family of description logics [Calvanese et al. 2005; 2006; 2008]
- Can exploit **query rewriting** based reasoning technique
  - Computationally optimal
  - Data storage and query evaluation can be delegated to standard RDBMS
  - Novel technique to prevent exponential blowup produced by rewritings [Kontchakov et al. 2010, Rosati and Almatelli 2010]
  - Can be extended to more expressive languages (beyond AC^0) by delegating query answering to a Datalog engine [Perez-Urbina et al. 2009]

Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $\mathcal{Q}$, use $\mathcal{O}$ to rewrite $\mathcal{Q}$ as $\mathcal{Q}'$ s.t., for any set of ground facts $\mathcal{A}$:
  - $\text{ans}(\mathcal{Q}, \mathcal{O}, \mathcal{A}) = \text{ans}(\mathcal{Q}', \emptyset, \mathcal{A})$
- Use (GAV) mapping $\mathcal{M}$ to map $\mathcal{Q}'$ to SQL query



Query Rewriting Technique (basics)

- Given ontology $\mathcal{O}$ and query $\mathcal{Q}$, use $\mathcal{O}$ to rewrite $\mathcal{Q}$ as $\mathcal{Q}'$ s.t., for any set of ground facts $\mathcal{A}$:
  - ans($\mathcal{Q}$, $\mathcal{O}$, $\mathcal{A}$) = ans($\mathcal{Q}'$, $\emptyset$, $\mathcal{A}$)
- Use (GAV) mapping $\mathcal{M}$ to map $\mathcal{Q}'$ to SQL query
- Resolution based query rewriting
  - **Clausify** ontology axioms
  - **Saturate** (clausified) ontology and query using resolution
  - **Prune** redundant query clauses

Query Rewriting Technique (basics)

- Example:

$$\text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient}$$

$$\text{Consultant} \sqsubseteq \text{Doctor}$$

$$Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$

$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$

$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$

$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$

$$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$
$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$
$$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$$
$$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$
$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$
$$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$$
$$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$$
$$Q(x) \leftarrow \mathsf{Doctor}(x)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$
$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$
$$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$$
$$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$$
$$Q(x) \leftarrow \mathsf{Doctor}(x)$$
$$Q(x) \leftarrow \mathsf{Consultant}(x)$$

Query Rewriting Technique (basics)

- Example:

$$\mathsf{Doctor} \sqsubseteq \exists \mathsf{treats}.\mathsf{Patient}$$
$$\mathsf{Consultant} \sqsubseteq \mathsf{Doctor}$$

$$\mathsf{treats}(x, f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Patient}(f(x)) \leftarrow \mathsf{Doctor}(x)$$
$$\mathsf{Doctor}(x) \leftarrow \mathsf{Consultant}(x)$$

$$Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$
~~$Q(x) \leftarrow \mathsf{Doctor}(x) \wedge \mathsf{Patient}(f(x))$~~
~~$Q(x) \leftarrow \mathsf{treats}(x, f(x)) \wedge \mathsf{Doctor}(x)$~~
$$Q(x) \leftarrow \mathsf{Doctor}(x)$$
$$Q(x) \leftarrow \mathsf{Consultant}(x)$$

- For DL-Lite, result is a union of conjunctive queries

$$Q(x) \leftarrow (\mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)) \vee \mathsf{Doctor}(x) \vee \mathsf{Consultant}(x)$$

Query Rewriting Technique (basics)

- UCQ translated into **SQL query** → OWL2QL can be "delegated" to RDBMS:

$$Q(x) \leftarrow (\mathsf{treats}(x,y) \wedge \mathsf{Patient}(y)) \vee \mathsf{Doctor}(x) \vee \mathsf{Consultant}(x)$$

$$\rightsquigarrow$$

SELECT Name FROM Doctor UNION
SELECT DName FROM Treats, Patient WHERE PName=Name

Interplay OWL2 ←→ RIF ←→ SPARQL1.1

- OWL2 and RIF
  - RIF fly-over
  - OWL2RL in RIF
  - RIF/OWL joint interpretations and what you need to know about them
- OWL2 and SPARQL1.1
  - SPARQL Entailment Regimes
  - Challenges+Pitfalls
  - What's in the current SPARQL 1.1 Draft?
- GiaBATA
  - A prototype implementation of SPARQL with dynamic Entailment regimes (e.g. RDFS, OWL2RL).

RIF fly-over

OWL and RIF

- RIF: Rule Interchange Format (rather than Rule language)
  - Framework for Rule Languages
  - Support RDF import: interesting for rule languages on top of RDF
  - Built-Ins support (close to XPath/XQuery functions and operators)
  - RIF Dialects:
    - RIF BLD: basic logic dialect = Horn rules with Built-ins, Equality
    - RIF Core: Datalog fragment (no logical function symbols, no head-equality)
    - RIF PRD: Production rules dialect
  - Normative XML syntax

- Commonalities with OWL:
  - RIF can model OWL2 RL
  - Share same Datatypes (XSD Datatypes, most OWL2 Datatypes)

- Differences
  - Different target audience: E.g. production rules (RIF PRD dialect)
  - Not necessarily focused on decidability, BLD = generic HORN rules with built-ins and function symbols (Turing-complete language)

RIF Dialects

Core

- horn rules, monotonic
- datatypes & built-ins
- external functions
- Frames, class memberships
- equality (in conditions)
- ground lists
- existential quantification (in conditions)

BLD

- equality, class membership in conclusions
- frame subclasses
- open lists

PRD

- non-monotonic
- actions in conclusions
- negation
- subclasses
- membership in conclusion

Example – RIF Core

- Full name in FOAF from givenName, familyName

```
if ({ ?X a foaf:Person ; foaf:givenName ?F ; foaf:familyName ?S } AND
     ?N = fn:concat(?F, " ", ?S) )
then { ?F foaf:name ?N }
```

  - Not expressible in OWL2, neither in SPARQL1.0 CONSTRUCT

```
CONSTRUCT { ?X foaf:name ?N }
WHERE {?X a foaf:Person; foaf:givenName ?F ; foaf:familyName ?S
          FILTER (?N = fn:concat(?F, " ", ?S)) }
```

Example – RIF Core

- Full name in FOAF from givenName, familyName

```
{ ?F foaf:name ?N } :-
{ ?X foaf:givenName ?F ; foaf:familyName ?S } AND
     ?N = fn:concat(?F, " ", ?S)
```

  - Can be read like Logic Programming rule
  - Presentation syntax not normative, we use a Mix of N3 and non-normative Presentation syntax in the spec here.

Example – RIF Core

- Full name in FOAF from givenName, familyName

```
?F[->foaf:name ?N]  :-
        ?X[foaf:givenName ?F], ?X[foaf:familyName->?S],
        ?N = fn:concat(?F, " ", ?S) .
```

- Can be read like Logic Programming rule
- Presentation syntax not normative, we use a Mix of N3 and non-normative Presentation syntax in the spec here.
  - RIF has F-Logic style Frames (e.g. FLORA-2)… same semantics as RDF-Triples
  - Further in rif # corresponds to class membership, ## to subclassing
  - in combination with RDF, # is the same as rdf:type)

OÉ Gaillimh NUI Galway

science foundation ireland fondúireacht eolaíochta éireann

DERI

Example – RIF BLD

- ATTENTION: Class membership # in conclusions is not in RIF Core.

```
{ ?x rdf:type ?y } :- ?x # ?y
 ?x # ?y :- { ?x rdf:type ?y }
```

Translating OWL2RL into RIF

OWL 2RL can be rewritten to RIF

- http://www.w3.org/TR/rif-owl-rl/

- Translates OWL2RL profile into RIF, relatively straightforward translation of abstract rules from
http://www.w3.org/TR/owl2-profiles/#Reasoning_in_OWL_2_RL_and_RDF_Graphs_using_Rules
  - Appendix 7: Static ruleset

  - Appendix 8: Dynamically instantiating a RIF Core rule set for a given OWL 2 RL, similar in spirit to the embedding in http://www.w3.org/TR/rif-rdf-owl/ Section 9.2

Static ruleset

- Some rules straightforward,e.g.

```
prp-ifp:  { ?S1 owl:SameAs ?S2 } :-
            { ?S1 ?P ?O . ?S2 ?P ?O . ?P rdf:type owl:InverseFunctionalProperty }

eq-rep-s: { ?Y ?P ?O } :- { ?X owl:SameAs ?Y . ?X ?P ?O }
eq-rep-p: { ?S ?Y ?O } :- { ?X owl:SameAs ?Y . ?S ?X ?O }
eq-rep-o: { ?S ?P ?Y } :- { ?X owl:SameAs ?Y . ?S ?P ?X }
```

- Others need auxiliary predicates for the static version:

prp-spo2: **{?S ?P ?O$_{n+1}$} :- { ?P owl:propertyChainAxiom ?pc }**
AND _checkChain(?S ?pc ?O$_{n+1}$)

We’d need that rule for all n, i.e. different property chain lenghts appearing in the ontology at hand.

Static ruleset:

prp-spo2: **Can be handled with auxiliary predicates:**

```
{ ?S ?P ?On+1 } :-  { ?P owl:propertyChainAxiom ?pc }
                  AND _checkChain(?S ?pc ?On+1)

 _checkChain(?start ?pc  ?last) :-
      { ?pc rdf:first ?p ; rdf:rest rdf:nil . ?start ?p ?last }

_checkChain(?start ?pc  ?last) :- And (
      { ?pc rdf:first ?p ; rdf:rest ?tl . ?start ?p ?next }
      _checkChain(?next ?tl ?last) ))
```

Other rules, e.g. subclassOf, inverseOf

cax-sco:

```
{ ?S rdf:type ?D } :-  { ?S rdf:type ?C . ?C rdfs:subclassOf ?D }
```

prp-inv1:

```
{ ?O ?P2 ?S } :-  { ?P1 inverseOf ?P2 . ?S ?P1 ?O }
```

prp-inv2:

```
{ ?O ?P1 ?S } :-  { ?P1 inverseOf ?P2 . ?S ?P2 ?O }
```

Similarly for other rules:

- all of OWL2RL can be translated to RIF Core rules, fed into your favorite rules engine, used for
  - Query answering,
  - Consistency checking

Dynamic ruleset (Template-Rules)

- Idea:
  - Translate each ontology **axiom by axiom** dynamically
  - E.g. ontology in RDF

```
foaf:Person rdfs:subClassOf foaf:Agent

foaf:topic owl:inverseOf foaf:page
```

Dynamic ruleset (Template-Rules)

- Idea:
  - Translate each ontology **axiom by axiom** dynamically
  - E.g. ontology in RDF
- Matching Template Rules in Appendix 8.2:

```
{?x rdf:type foaf:Agent} :- {?x rdf:type foaf:Person }

{ ?Y foaf:topic ?X } :- { ?Y foaf:page ?X }
{ ?Y foaf:page ?X }  :- { ?Y foaf:topic ?X }
```

Plus some fixed ruleset (Appendix 8.1 FixedRules in http://www.w3.org/TR/rif-owl-rl/

Embedding OWL2RL Ontologies into RIF

for combinations with arbitrary RIF rulesets

RIF + RDF and OWL in combination

- RIF/OWL joint interpretations
  - http://www.w3.org/TR/rif-rdf-owl/ defines semantic correspondence between RIF and RDF/RDFS/OWL interpretations,
- i.e., semantics for combinations of RDF graphs, OWL ontologies and RIF rulesets
- Defines:
  - ***RIF-OWL-Direct Entailment:*** *Based on OWL direct semantics*

    RIF-OWL-DL combination disallows certain RIF documents

    (only constants for classes in #, ##, only constants for predicates in frames) …
  - ***RIF-OWL RDF-Based Entailment:***

    Based on OWL RDF-Based Semantics.

Embedding

- (Informative) Embedding in http://www.w3.org/TR/rif-rdf-owl/ give rise for implementation of combination of OWL2RL and RIF :

1. Embedding RIF DL-document formulas into RIF BLD, , Section 9.2.1
2. **Embedding OWL 2 RL axioms into RIF BLD, Section 9.2.2**

We focus on the latter part…

Embedding OWL2RL axioms into rules:

- Section 9.2.2 defines recursive translation from OWL axioms to RIF rules… tr()
- Very similar to Dynamic Rules we saw before
- E.g. OWL2RL ontology in RDF

```
foaf:Person rdfs:subClassOf foaf:Agent
```

```
foaf:topic owl:inverseOf foaf:page
foaf:topic type owl:ObjectProperty
foaf:page type owl:ObjectProperty
```

Embedding OWL2RL axioms into rules:

- Section 9.2.2 defines recursive translation from OWL axioms to RIF rules… tr()
- Very similar to Dynamic Rules we saw before
- Translated to OWL abstract syntax axioms:

```
SubClassOf(foaf:Person foaf:Agent)

InverseObjectProperties(foaf:topic foaf:page)
```

Embedding OWL2RL axioms into rules:

- Section 9.2.2 defines recursive translation from OWL axioms to RIF rules… tr()
- Very similar to Dynamic Rules we saw before
- Translated to RIF by translation tr() in Section 9.2.2 of http://www.w3.org/TR/rif-rdf-owl/ :

```
{?x rdf:type foaf:Agent} :- {?x rdf:type foaf:Person }

{ ?Y foaf:topic ?X } :- { ?Y foaf:page ?X }
{ ?Y foaf:page ?X }  :- { ?Y foaf:topic ?X }
```

Plus some static ruleset ($R^{OWL\text{-}Direct}(V,R)$ ***)***

OÉ Gaillimh
NUI Galway

science foundation ireland
fondúireacht eolaíochta éireann

DERI

Subtle differences to direct OWL2RL translation from before:

- Most fundamentally **equality**: owl:sameAs is directly translated to RIF's =, rather than axiomatised as in slide 54:

- OWL RDF:

```
<http://a> owl:sameAs <http://b> .
```

  is embedded as:

```
<http://a> = <http://b>
```

- E.g. in Combination with RIF ruleset:

```
_q(<http://a>).
_p(?x) :- iri-to-string(?y, ?x) and _q(?y)
```

  entails:

```
_p("http://a").
_p("<http://b>").
```

- Not so if I take the axiomatisation of sameAs from above
- ***Bottomline: To straightforwardly implement the embedding for combinations, You need a rule system that supports equality.***

SPARQL 1.1 querying over OWL2 ontologies?

OWL2 and SPARQL1.1

- SPARQL1.1 working group will define SPARQL query answering over OWL2 ontologies and RIF rule sets:
  - http://www.w3.org/TR/sparql11-entailment/
- Latest Working Draft just released…
  - Contains Draft Semantics for
    - SPARQL1.1 on top of RDFS
    - **SPARQL1.1 on top of OWL2**
    - SPARQL1.1 on top of RIF

OWL2 and SPARQL1.1

- General Idea: Answer Queries with implicit answers
- E.g. Graph

```
foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Person rdfs:subclassOf
     [ a owl:Restriction ;
      owl:onProperty :hasFather ;
      owl:someValuesFrom foaf:Person. ]
:jeff a Person
:jeff foaf:knows :aidan
foaf:knows rdfs:range foaf:Person.
```

`SELECT ?X { ?X a foaf:Person }`

Pure SPARQL 1.0 returns only `:Jeff`,
should also return `:aidan`

SPARQL+RDFS/OWL: Challenges+Pitfalls

- Challenges+Pitfalls:
  - Possibly Infinite answers (by RDFS ContainerMembership properties, OWL datatype reasoning, etc.)
  - Conjunctive Queries: non-distinguished variables
  - SPARQL 1.1 features: Aggregates

SPARQL+RDFS/OWL: Challenges+Pitfalls

- Current Solution:
  - Possibly Infinite answers (by RDFS ContainerMembership properties, OWL datatype reasoning, etc.)
    - Restrict answers to rdf:/rdfs:/owl:vocabulary minus rdf:_1 … rdf:_n plus terms occurring in the data graph
  - Non-distinguished variables
    - *No non-distinguished variables, answers must result from BGP matching, projection a post-processing step not part of entailment.*
  - SPARQL 1.1 other features: Aggregates
    - *Again not affected, answers must result from BGP matching, projection a post-processing step not part of entailment.*
  - Simple, BUT: maybe not always entirelty intuitive, so
    - Good to know ;-)

Possibly Infinite answers RDF(S): Container Membership

- Graph:

```
:me :hasFavouritePresenter [ a rdf:Seq;
                                    rdf:_1 :jeff.
                                    rdf:_2 :aidan.
                                    rdf:_3 :axel.  ]
```

Query with RDFS Entailment in mind:

`SELECT ?CM { ?CM a rdfs:ContainerMembershipProperty}`

Entailed by RDFS (axiomatic Triples):

```
rdfs:_1 a rdfs:ContainerMembershipProperty .
rdfs:_2 a rdfs:ContainerMembershipProperty .
rdfs:_3 a rdfs:ContainerMembershipProperty .
rdfs:_4 a rdfs:ContainerMembershipProperty .
…
```

Possibly Infinite answers RDF(S): Container Membership

- Graph:

```
:me :hasFavouritePresenter [ a rdf:Seq;
                               rdf:_1 :jeff.
                               rdf:_2 :aidan.
                               rdf:_3 :axel.  ]
```

Query with RDFS Entailment in mind:

`SELECT ?CM { ?CM a rdfs:ContainerMembershipProperty}`

SPARQL 1.1 restricts answers to rdf:/rdfs:/owl:vocabulary minus rdf:_1 … rdf:_n **plus terms occurring in the data graph**

So, the only answers are:

```
{ ?CM/rdfs:_1, ?CM/rdfs:_2, ?CM/rdfs:_3 }
```

Possibly Infinite answers OWL: datatype reasoning

Stupid way to say Peter is 50:

```
ex:Peter a [ a owl:Restriction ;
             owl:onProperty ex:age ;
             owl:allValuesFrom [ rdf:type rdfs:Datatype .
             owl:oneOf ("50"^^xsd:integer) ] ]
```

Stupid query asking What is NOT Peters age:

```
SELECT ?x WHERE {
  ex:Peter a [ a owl:Restriction ; owl:onProperty ex:age ;
       owl:allValuesFrom [ a rdfs:Datatype ;
                                   owl:datatypeComplementOf [ a
                  rdfs:Datatype ; owl:oneOf (?x) ] ] ] }
```

Theoretical answer: all literal different from 50

No danger in SPARQL 1.1 restricts answers to rdf:/rdfs:/owl:vocabulary minus rdf:_1 … rdf:_n **plus terms occurring in the data graph**

Non-distinguished variables:

- E.g. Graph

```
 foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Person rdfs:subclassOf
         [ a owl:Restriction ;
           owl:onProperty :hasFather ;
           owl:someValuesFrom foaf:Person. ]
:jeff a Person
:jeff foaf:knows :aidan
foaf:knows rdfs:range foaf:Person.
```

SELECT ?X ?Y { ?X :hasFather ?Y }

No answer, because no known value for ?Y in the data graph.

Non-distinguished variables:

- E.g. Graph

```
 foaf:Person rdfs:subClassOf foaf:Agent .
foaf:Person rdfs:subclassOf
         [ a owl:Restriction ;
           owl:onProperty :hasFather ;
           owl:someValuesFrom foaf:Person. ]
:jeff a Person
:jeff foaf:knows :aidan
foaf:knows rdfs:range foaf:Person.
```

SELECT ?X { ?X :hasFather ?Y }

But what about this one? ?Y looks like a "non-distinguished" variable

Solution: In SPARQL 1.1 answers must result from BGP matching, projection a post-processing step not part of entailment ➔ so, still no answer.

Non-distinguished variables:

- Simple Solution may seem not always intuitive, but:
  - OWL Entailment in SPARQL based on BGP matching, i.e.
    - always only returns results with named individuals
    - Doesn't affect SELECT: takes place before projection
    - That is: **non-distinguished variables can't occur "by design"**
  - In fact, conjunctive queries with non-distinguished variable still an open research problem for OWL:
    - Decidable for SHIQ, [B. Glimm et al. 2008]
    - Decidable for OWL1 DL without transitive properties OWL1 Lite without nominals [B. Glimm, KR-10]
    - Decidability for SHOIN, SROIQ though still unknown…

SPARQL 1.1 other features: Aggregates

- Similar as before… aggregates are evaluated as post-processing **after** BGP matching, so, no effect:

```
foaf:Person rdfs:subClassOf foaf:Agent .
  foaf:Person rdfs:subclassOf
        [ a owl:Restriction ;
        owl:onProperty :hasFather ;
        owl:someValuesFrom foaf:Person. ]
  :jeff a Person
  :jeff foaf:knows :aidan
  foaf:knows rdfs:range foaf:Person.

SELECT ?X { ?X a foaf:Person }
```

Under RDFS/OWL entailment returns : {?X/jeff, ?X/aidan}

SPARQL 1.1 other features: Aggregates

- Similar as before… aggregates are evaluated as post-processing **after** BGP matching, so, no effect:

```
foaf:Person rdfs:subClassOf foaf:Agent .
  foaf:Person rdfs:subclassOf
        [ a owl:Restriction ;
        owl:onProperty :hasFather ;
        owl:someValuesFrom foaf:Person. ]
  :jeff a Person
  :jeff foaf:knows :aidan
  foaf:knows rdfs:range foaf:Person.
```

```
SELECT ?Y AS Count(?X)  { ?X a foaf:Person }
```

Under RDFS/OWL entailment returns : {?Y/2}

GiaBATA

Implementing SPARQL, OWL2RL, RIF on top of DLV

GiaBATA

- Time allowed, we will show a system which implements dynamic SPARQL querying, under different entailment regimes and how it can be implemented.

- Based on LP engine DLV
  - Datalog with built-ins (covers roughly RIF Core),
  - persistent Database backend (DLV-DB)
  - Optimisations (rewriting to push join processing into SQL as far as possible, magic sets,…)
  - plus a lot more features (nonmonotonicity, aggregates, …)

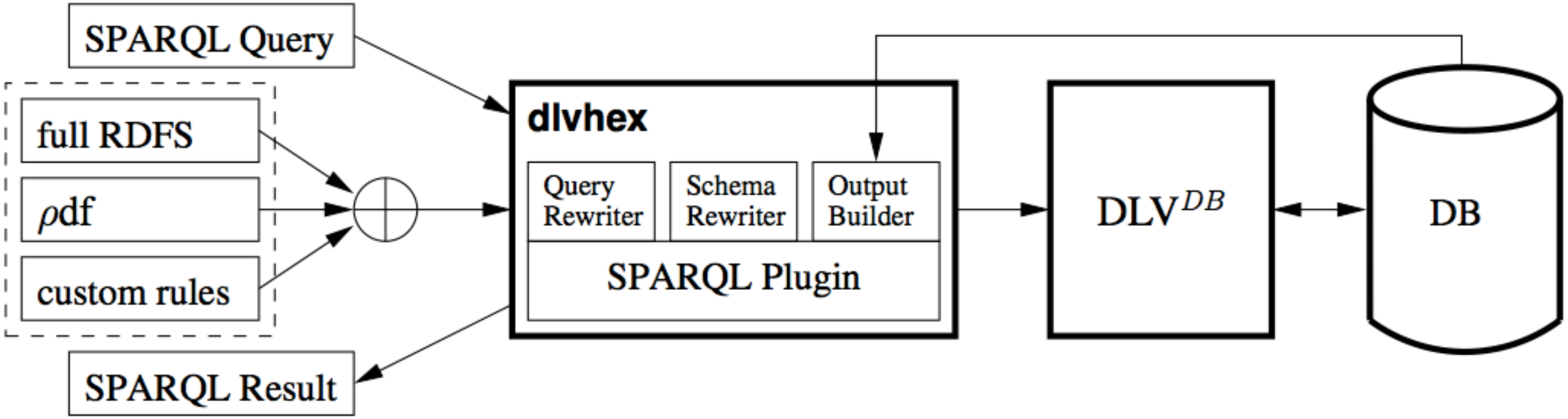
- Overall idea for SPARQL+RDFS/OWL2RL over RDF graphs:
  - Translate OWL2RL to Datalog rules a la RIF, see above.
  - Translate SPARQL query to Datalog [Polleres, WWW2007]
  - Feed resulting program into a rules engine (DLV-DB) that runs over a Rel DB storing RDF graphs.

- Check Details at:
- http://axel.deri.ie/~axepol/presentations/20091029iann-etal-ISWC2009_GiaBATA.pptx

How to implement this?

- GiaBATA system [Ianni et al., 2009]:
  - SPARQL → dlvhex (logic program)
  - Ruleset → 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

SPARQL → dlvhex (logic program)

- Based on [Polleres ,WWW2007]

```
select * from <http://alice.org/>
 where { ?X a foaf:Person. ?X foaf:name ?N.
         filter ( ?N != "Alice") optional { ?X foaf:mbox ?M } }
```

```
(r1) "triple"(S,P,O,default)     :- &rdf[ "alice.org" ](S,P,O).
(r2) answer1(X_N,X_X,default)    :- "triple"(X_X,"rdf:type","foaf:Person",default),
                                    "triple"(X_X,"foaf:name",X_N,default),
                                    &eval[" ?N != 'Alice' ","N", X_N ](true).
(r3) answer2(X_M,X_X,default)    :- "triple"(X_X,"foaf:mbox",X_M,default).
(r4) answer_b_join_1(X_M,X_N,X_X,default)   :- answer1(X_N,X_X,default),
                                               answer2(X_M,X_X,default).
(r5) answer_b_join_1(null,X_N,X_X,default) :- answer1(X_N,X_X,default),
                                               not answer2_prime(X_X,default).
(r6) answer2_prime(X_X,default) :- answer1(X_N,X_X,default),
                                    answer2(X_M,X_X,default).
(r7) answer(X_M,X_N,X_X)         :- answer_b_join1(X_M,X_N,X_X,default).
```

OWL2RL Static Ruleset → dlvhex (logic program)

- Straighforward, just translates rules in a way “compatible” with the SPARQL translation:

```
{?s ?q ?o } <= {?s ?p ?o . ?p rdfs:subPropertyOf ?q}
```

```
%FROM CLAUSES
triple(P,SubPropertyOf,P,G)  :- triple(P,Type,Property,G),graph(G,D),data(D),defaultGraph(D),
                                resource_literal(Type,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>",_),
                                resource_literal(Property,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>",_),
                                resource_literal(SubPropertyOf,"<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>",_).

%FROM NAMED CLAUSES
triple(P,SubPropertyOf,P,G)  :- triple(P,Type,Property,G),graph(G,D),data(D),namedGraph(D),
                                resource_literal(Type,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>",G),
                                resource_literal(Property,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>",G),
                                resource_literal(SubPropertyOf,"<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>",G).

%USING ONTOLOGIES
triple(P,SubPropertyOf,P,G) :- triple(P,Type,Property,G),graph(G,D),data(D),ontology(D),
                               resource_literal(Type,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>",G),
                               resource_literal(Property,"<http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>",G),
                               resource_literal(SubPropertyOf,"<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>",G).
```

SPARQL+Rules → SQL

- Done by dlv-DB, cf. [Terracina, et al.,2008]
  - All non-recursive parts are pushed to the Database
  - All recursive parts handled by semi-naïve evaluation

    (more efficient than WITH RECURSIVE views in SQL, where necessary, intermediate results temporarily materialized into the DB)

- Some necessary optimisations to make this *reasonably* performant:
  - FILTER expression evaluation is pushed to SQL (3-valued semantics of SPARQL Filters is handled natively in SQL)
  - No miracles… but magic: Magic set optimisations for focused fwd-chaining evaluation.
  - Join-reordering, not yet implemented, but we did some manual reordering to optimize the query plan in the experiments.

References OWL2, Scalable/Tractable OWL Reasoning:

[Baader et al. 2005] Franz Baader, Sebastian Brandt, Carsten Lutz: **Pushing the EL Envelope**. IJCAI 2005: 364-369

[Calvanese et al. 2005] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati: **DL-Lite: Tractable Description Logics for Ontologies**. AAAI 2005: 602-607

[Calvanese et al. 2006] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati: **Data Complexity of Query Answering in Description Logics**. KR 2006: 260-270

[Calvanese et al. 2008] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati: **View-Based Query Answering over Description Logic Ontologies**. KR 2008: 242-251

[Glimm et al. 2008] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. **Conjunctive query answering for the description logic SHIQ**. J. Artif. Intell. Res. (JAIR), 31:157–204, 2008.

[Birte Glimm and Sebastian Rudolph]. Status QIO: **Conjunctive Query Entailment is Decidable**. In Principles of Knowledge Representation and Reasoning (KR 2010), 2010.

[Delbru et al. 2008] R. Delbru, A. Polleres, G. Tummarello and S. Decker: **Context Dependent Reasoning for Semantic Documents in Sindice**. In *Proceedings of the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems*.

[Hogan et al. 2008] Aidan Hogan, Andreas Harth, Axel Polleres: **SAOR: Authoritative Reasoning for the Web**. In *Proceedings of the 3rd Asian Semantic Web Conference.*

[Hogan&Decker 2009] Aidan Hogan, Stefan Decker: **On the Ostensibly Silent 'W' in OWL 2 RL**. In *Proceedings of the 3rd International Conference on Web Reasoning and Rule Systems*.

[Hogan et al. 2009] Aidan Hogan, Andreas Harth, Axel Polleres: **Scalable Authoritative OWL Reasoning for the Web**. In *International Journal on Semantic Web and Information Systems (IJSWIS)*.

[Motik 2007] Boris Motik: **On the Properties of Metamodeling in OWL**. J. Log. Comput. 17(4): 617-637 (2007).

[Motik and Horrocks 2008] Boris Motik and Ian Horrocks. **OWL Datatypes: Design and Implementation**. In Proc. of the 7th International Semantic Web Conference (ISWC 2008), volume 5318 of Lecture Notes in Computer Science, pages 307-322. Springer, 2008.

[Horrocks et al. 2006] Ian Horrocks, Oliver Kutz, Ulrike Sattler: **The Even More Irresistible SROIQ**. KR 2006: 57-67.

[Kazakov 2009] Yevgeny Kazakov: **Consequence-Driven Reasoning for Horn SHIQ Ontologies**. IJCAI 2009: 2040-2045

References OWL2, Scalable/Tractable OWL Reasoning:

[Kontchakov et al. 2010] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter and Michael Zakharyaschev. **The Combined Approach to Query Answering in DL-Lite**. In Proc. of KR2010.

[Ortiz et al. 2006] Magdalena Ortiz, Diego Calvanese, Thomas Eiter: **Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics**. AAAI 2006

[Pan 2004] Jeff Z. Pan. **Description Logics: Reasoning Support for the Semantic Web**. Ph.D. Thesis, School of Computer Science, the University of Manchester, 2004.

[Pan and Horrocks 2006] Jeff Z. Pan and Ian Horrocks. **OWL-Eu: Adding Customised Datatypes into OWL**. In Journal of Web Semantics, 4(1). 29-49. 2006.

[Pan, Horrocks, Schreiber, 2005] Jeff Z. Pan, Ian Horrocks and Guus Schreiber. **OWL FA: A Metamodeling Extension of OWL DL**. In Proc. of the International workshop on OWL: Experience and Directions (OWL-ED2005). 2005

[Pan and Thomas, 2007] Jeff Z. Pan and Edward Thomas. **Approximating OWL-DL Ontologies**. In Proc. of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07). 1434-1439. 2007.

[Pan et al. 2009] Jeff Z. Pan, Edward Thomas and Yuting Zhao. **Completeness Guaranteed Approximation for OWL DL Query Answering**. In Proc. of the 22nd International Workshop on Description Logics (DL2009). 2009.

[Pérez-Urbina et al. 2010] Héctor Pérez-Urbina, Boris Motik, Ian Horrocks: **Tractable query answering and rewriting under description logic constraints**. J. Applied Logic 8(2): 186-209 (2010)

[Ren et al. 2010] Yuan Ren, Jeff Z. Pan and Yuting Zhao. **Soundness Preserving Approximation for TBox Reasoning**. In Proc. of the 25th AAAI Conference Conference (AAAI2010). 2010.

[Rosati and Almatelli 2010] Riccardo Rosati and Alessandro Almatelli. **Improving query answering over DL-Lite ontologies**. In Proc. of KR2010.

[ter Horst 2005] Herman J. ter Horst: **Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity**. In *Proceedings of the 4th International Semantic Web Conference*

[Urbani et al. 2009] Jacopo Urbani, Spyros Kotoulas, Eyal Oren, Frank van Harmelen: **Scalable Distributed Reasoning Using MapReduce**. In *Proceedings of the 8th International Semantic Web Conference*

[Weaver et al. 2009] Jesse Weaver and James A. Hendler. **Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples**. In *Proceedings of the 8th International Semantic Web Conference*

References on RDF and SPARQL Querying

[Gutierrez et al. 2004] Gutierrez, C., Hurtado, C.A., Mendelzon, A.O.: **Foundations of semantic web databases.** In: PODS 2004, ACM (2004) 95–106

[Muñoz et al. 2007] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. **Minimal deductive systems for RDF**. In Proceedings of the 4th European Semantic Web Conference (ESWC2007), 2007.

[Meier 2008] Michael Meier. **Towards Rule-Based Minimization of RDF Graphs under Constraints.** In Proc. RR'08, volume 5341 of LNCS, pages 89–103. Springer, 2008.

[Pichler et al. 2010] Reinhard Pichler, Axel Polleres, Sebastian Skritek, and Stefan Woltran. **Redundancy elimination on rdf graphs in the presence of rules, constraints, and queries**. In Proc. RR'10, Bressanone, Italy, September 2010.

[Polleres 2007] Axel Polleres. **From SPARQL to rules (and back)**. In Proceedings of the 16th World Wide Web Conference (WWW2007), pages 787–796, Banff, Canada, May 2007. ACM Press

[Pérez et al, 2006] Pérez, J., Arenas, M., Gutierrez, C.: **Semantics and complexity of SPARQL**. ACM Trans. Database Syst. 34(3) (2009)

[Angles&Gutierrez, 2008]Angles, R., Gutierrez, C.: **The expressive power of SPARQL**. In: Proc. ISWC 2008. Volume 5318 of LNCS., Springer (2008) 114–129

[Ianni et al., 2009] G. Ianni, T. Krennwallner, A. Martello, A. Polleres. **Dynamic querying of mass-storage RDF data with rule-based entailment regimes**. In Proceedings of the 8th International Semantic Web Conference (ISWC 2009), October 2009. Springer.

[Terracina, et al.,2008] Terracina, G., Leone, N., Lio, V., Panetta, C.: **Experimenting with recursive queries in database and logic programming systems.** Theory Pract. Log. Program. 8(2) (2008) 129–165.

Relevant W3C Standard Specs

- RDF Semantics http://www.w3.org/TR/rdf-mt/

- OWL2 Web Ontology Language Primer http://www.w3.org/TR/owl2-primer/
- OWL2 Web Ontology Language Profiles http://www.w3.org/TR/owl2-profiles/

- SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/
- SPARQL1.1 Query Language for RDF (working draft) http://www.w3.org/TR/sparql11-query/
- SPARQL1.1 Entailment Regimes (working draft) http://www.w3.org/TR/sparql11-entailment/

- RIF Core Dialect http://www.w3.org/TR/rif-core/
- RIF Basic Logic Dialect http://www.w3.org/TR/rif-bld/
- RIF RDF and OWL compatibility http://www.w3.org/TR/rif-rdf-owl/

Other Tutorials

- Some more basic lectures&Tutorials on my Website: http://www.polleres.net/ e.g.
  - Semantic Web and ASP Tutorial ESWC2006
  - SPARQL Tutorial ESWC2007
  - Scalable OWL Reasoning Tutorial ESWC2010

- Also recommended:
  - Reasoning Web Summer Schools (since 2005), many good tutorials/slides:
    - http://reasoningweb.org/2005/
    - http://reasoningweb.org/2006/
    - http://reasoningweb.org/2007/
    - http://reasoningweb.org/2008/
    - http://reasoningweb.org/2009/
    - http://reasoningweb.org/2010/
- Linked Data Tutorial:
  - http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

Acknowledgements

- Jeff Pan
- Aidan Hogan
- Giovambattista Ianni
- Alessandra Martello
- Thomas Krennwallner
- The members of the W3C OWL2 WG)
- The members of the W3C RIF WG
- The members of the W3C SPARQL WG