



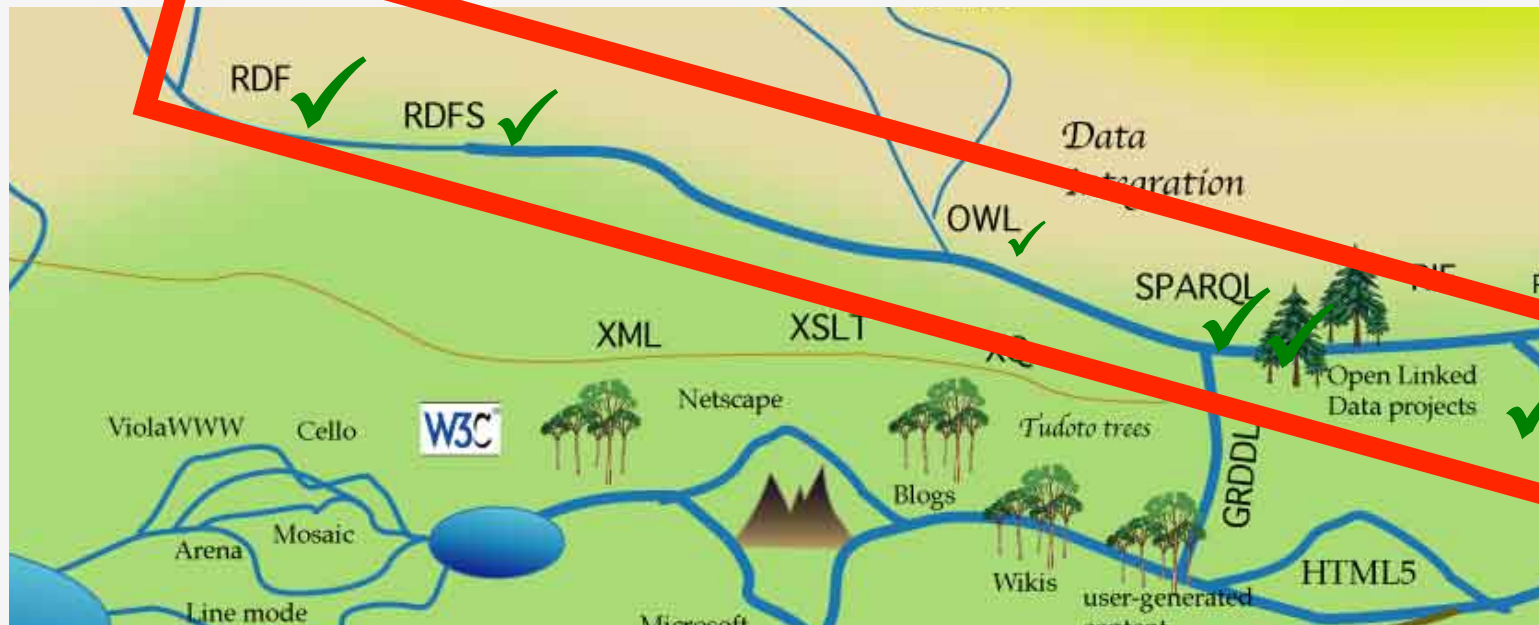
# Stream Reasoning For Linked Data

Jeff Z. Pan and Emanuele Della Valle



A portrait of Dr. Peter Dinklage, a middle-aged man with thinning brown hair, wearing a dark suit jacket over a blue shirt. He is looking directly at the camera with a slight smile. The background is a blurred interior with warm lighting and wooden paneling.

# The Web map 2008 © Tim Berners-Lee



- more and more ontologies and ontological stream data available thanks to
- ... related standards (RDF, OWL) becoming established
- ... the community (across many application domains) is growing fast
- ... In this tutorial: **How to use OWL 2 for linked data, as well as efficient and scalable stream reasoning for OWL 2**

# Agenda

- **Introduction to Linked Data and OWL 2 (90m)**
- C-SPARQL: A Continuous Extension of SPARQL (90m)
- Stream Reasoning techniques for RDFS and OWL2-RL (60m)
- Approximate Reasoning and Approximate Stream Reasoning for OWL2-DL (60m)
- Hands-on Session (60m)



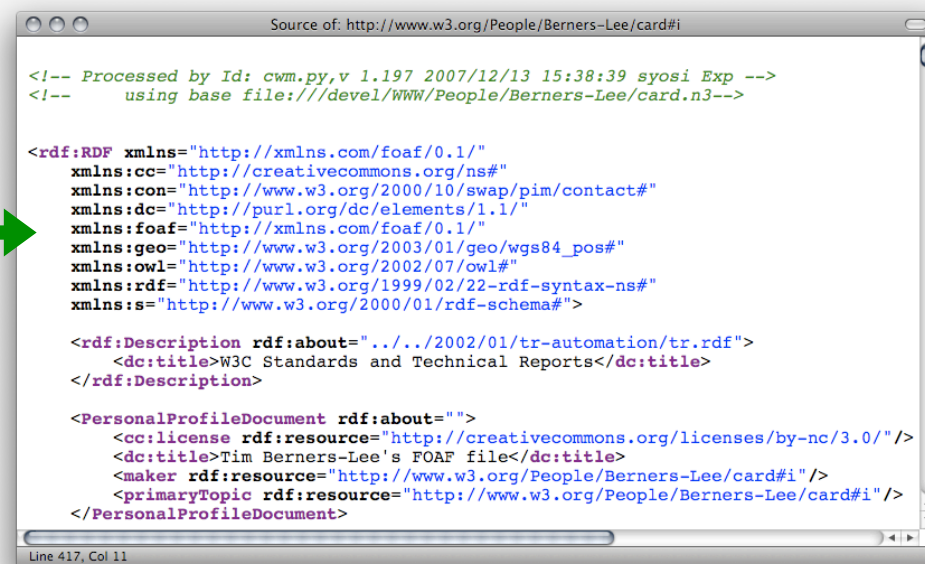
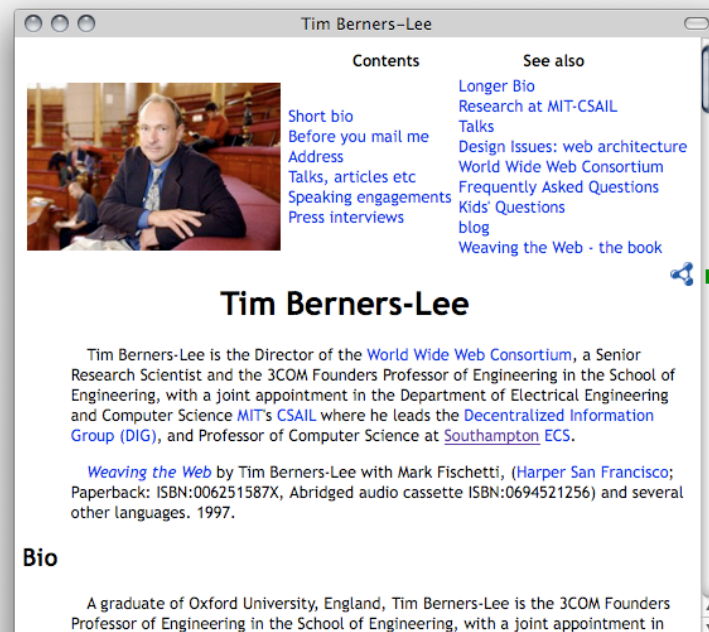
- Resource Description Framework
  - Makes content machine readable by introducing descriptions/annotations
- RDF statements
  - Subject-property-value triple
- Examples (in N3 and XML syntax)

```
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows  
<http://dblp.13s.de/d2r/.../Dan_Brickley> .
```

```
<rdf:Description rdf:about="http://www.w3.org/People/Berners-Lee/card#i">  
  <foaf:knows rdf:resource="http://dblp.13s.de/d2r/.../Dan_Brickley">  
</rdf:Description>
```

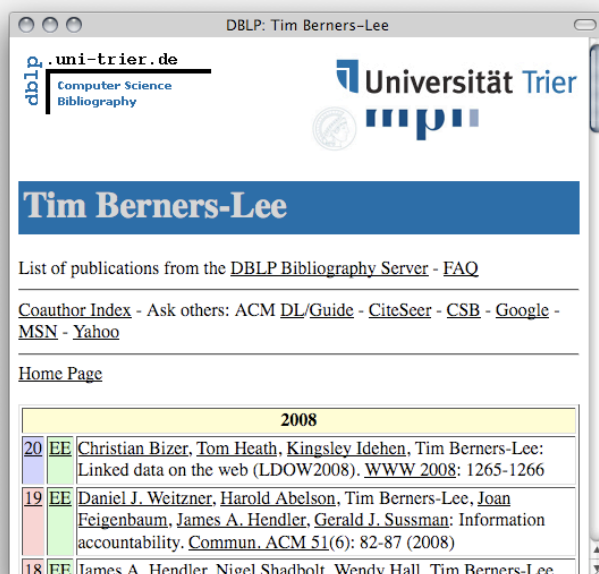
- (i) directly by the publishers
- (ii) by e.g. GRDDL transformations, D2R, RDFa exporters, etc.

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



- (i) directly by the publishers
- (ii) by e.g. GRDDL 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/authors/Tim_Berners-Lee),

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

Provides RDF version of all DBLP data + query interface!

# RDF Data online: Example

## ■ Data in RDF: **Triples**

- **DBLP:**

```
<http://dblp.13s.de/.../journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.  
<http://dblp.13s.de/.../journals/tplp/Berners-LeeCKSH08> dc:creator  
  <http://dblp.13s.de/d2r/.../Tim_Berners-Lee> .
```

...

```
<http://dblp.13s.de/d2r/.../Tim_Berners-Lee> foaf:homepage  
  <http://www.w3.org/People/Berners-Lee/> .
```

...

```
<http://dblp.13s.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.13s.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 that 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”
    - “Names of all persons who co-authored with authors of <http://dblp.l3s.de/d2r/.../Berners-LeeCKSH08> or known by co-authors”
    - ...

## 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 patterns: e.g. UNIONS

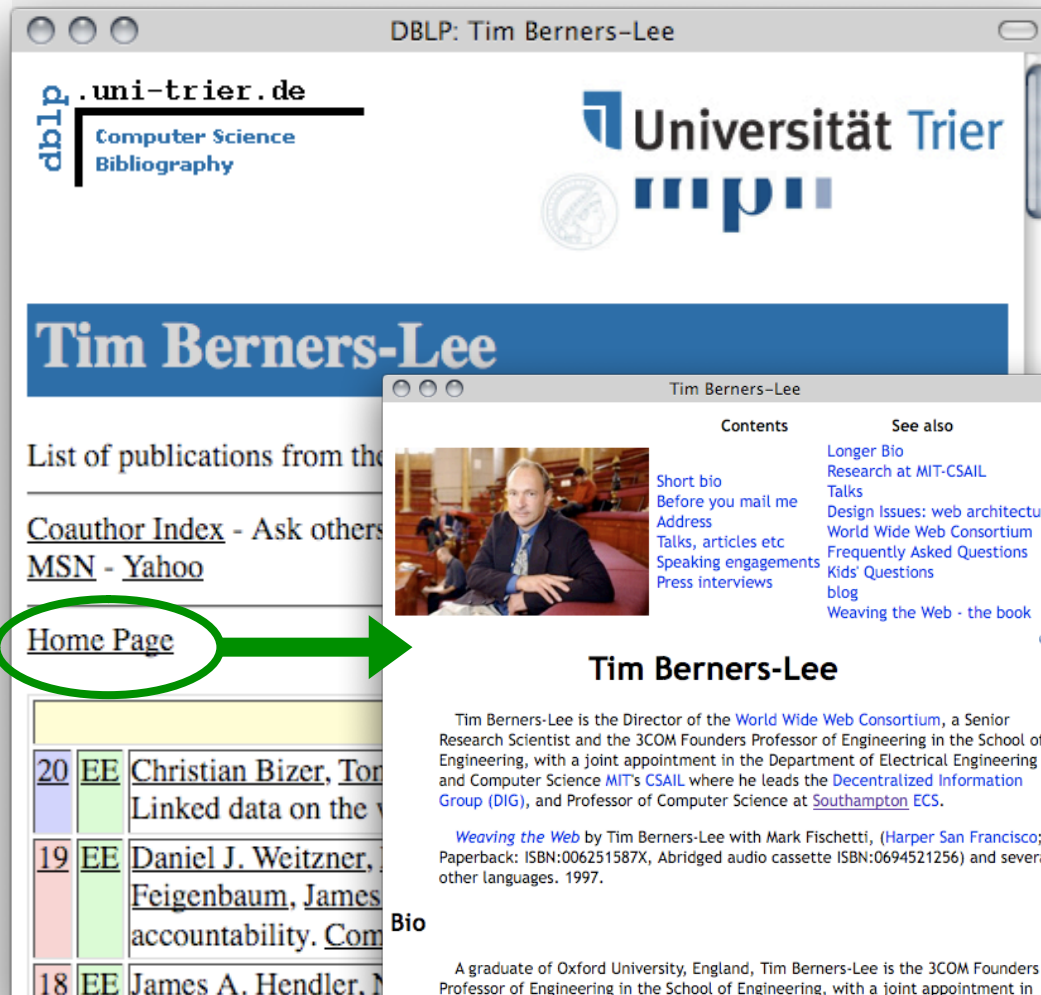
- *“Names of all persons who co-authored with authors of <http://dblp.13s.de/d2r/.../Berners-LeeCKSH08>”*

```
SELECT ?Name WHERE
```

```
{ <http://dblp.13s.de/d2r/resource/publication/journals/tplp/Berners-LeeCKSH08>
  dc:creator ?Author.
  ?D dc:creator ?Author.
  ?D dc:creator ?CoAuthor.
  ?CoAuthor foaf:name ?Name
}
```

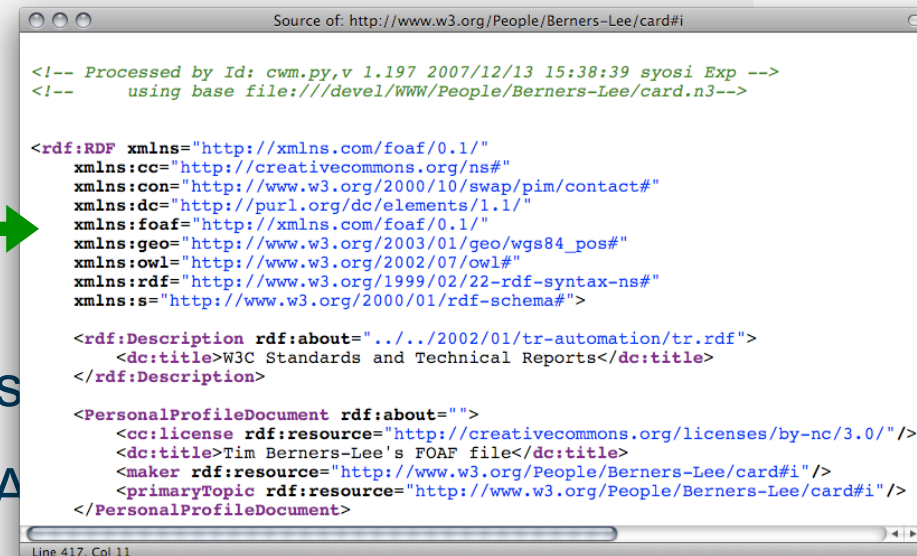
# SPARQL more complex patterns: e.g. UNIONS

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



The screenshot shows the DBLP: Tim Berners-Lee page. The top navigation bar includes links to [dblp.uni-trier.de](http://dblp.uni-trier.de), Computer Science, and Bibliography. The main header features the University of Trier logo and the name "Tim Berners-Lee". Below this, there is a "List of publications from the" section, a "Coauthor Index - Ask others" link, and a "Home Page" link circled in green. A green arrow points from the "Home Page" link to a smaller window showing a bio of Tim Berners-Lee. The bio window includes a photo, a "Contents" list (Short bio, Before you mail me, Address, Talks, articles etc, Speaking engagements, Press interviews), a "See also" list (Longer Bio, Research at MIT-CSAIL, Talks, Design Issues: web architecture, World Wide Web Consortium, Frequently Asked Questions, Kids' Questions, blog, Weaving the Web - the book), and a paragraph about his role as Director of the World Wide Web Consortium and Senior Research Scientist at MIT-CSAIL. Below the bio, there is a "Bio" section mentioning his graduation from Oxford University and his position as 3COM Founders Professor of Engineering.

[journals/tplp/Berners-LeeCKSH08](http://journals.tplp/Berners-LeeCKSH08)



The screenshot shows the source code of a FOAF file. The code is XML-based and includes various namespaces and properties. A green arrow points from the bio window to this code. The code includes the following elements:

```
<!-- Processed by Id: cwm.py,v 1.197 2007/12/13 15:38:39 syosi Exp -->
<!-- using base file:///devel/WWW/People/Berners-Lee/card.n3-->

<rdf:RDF xmlns="http://xmlns.com/foaf/0.1/"
  xmlns:cc="http://creativecommons.org/ns#"
  xmlns:con="http://www.w3.org/2000/10/swap/pim/contact#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://www.w3.org/2000/01/rdf-schema#"

  <rdf:Description rdf:about=".../2002/01/tr-automation/tr.rdf">
    <dc:title>W3C Standards and Technical Reports</dc:title>
  </rdf:Description>

  <PersonalProfileDocument rdf:about="">
    <cc:license rdf:resource="http://creativecommons.org/licenses/by-nc/3.0/" />
    <dc:title>Tim Berners-Lee's FOAF file</dc:title>
    <maker rdf:resource="http://www.w3.org/People/Berners-Lee/card#i" />
    <primaryTopic rdf:resource="http://www.w3.org/People/Berners-Lee/card#i" />
  </PersonalProfileDocument>
```

## Back to the Data:

- DBLP:

```
<http://dblp.13s.de/.../journals/tplp/Berners-LeeCKSH08> rdf:type swrc:Article.  
<http://dblp.13s.de/.../journals/tplp/Berners-LeeCKSH08> dc:creator  
  <http://dblp.13s.de/d2r/.../Tim_Berners-Lee> .
```

...

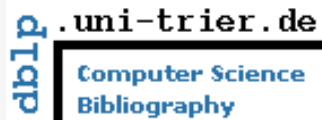
```
<http://dblp.13s.de/d2r/.../Tim_Berners-Lee> foaf:homepage  
  <http://www.w3.org/People/Berners-Lee/> .
```

- Tim Berners-Lee's FOAF file:

```
<http://www.w3.org/People/Berners-Lee/card#i> foaf:knows  
  <http://dblp.13s.de/d2r/.../Dan_Brickley> .  
<http://www.w3.org/People/Berners-Lee/card#i> foaf:homepage  
  <http://www.w3.org/People/Berners-Lee/> .
```

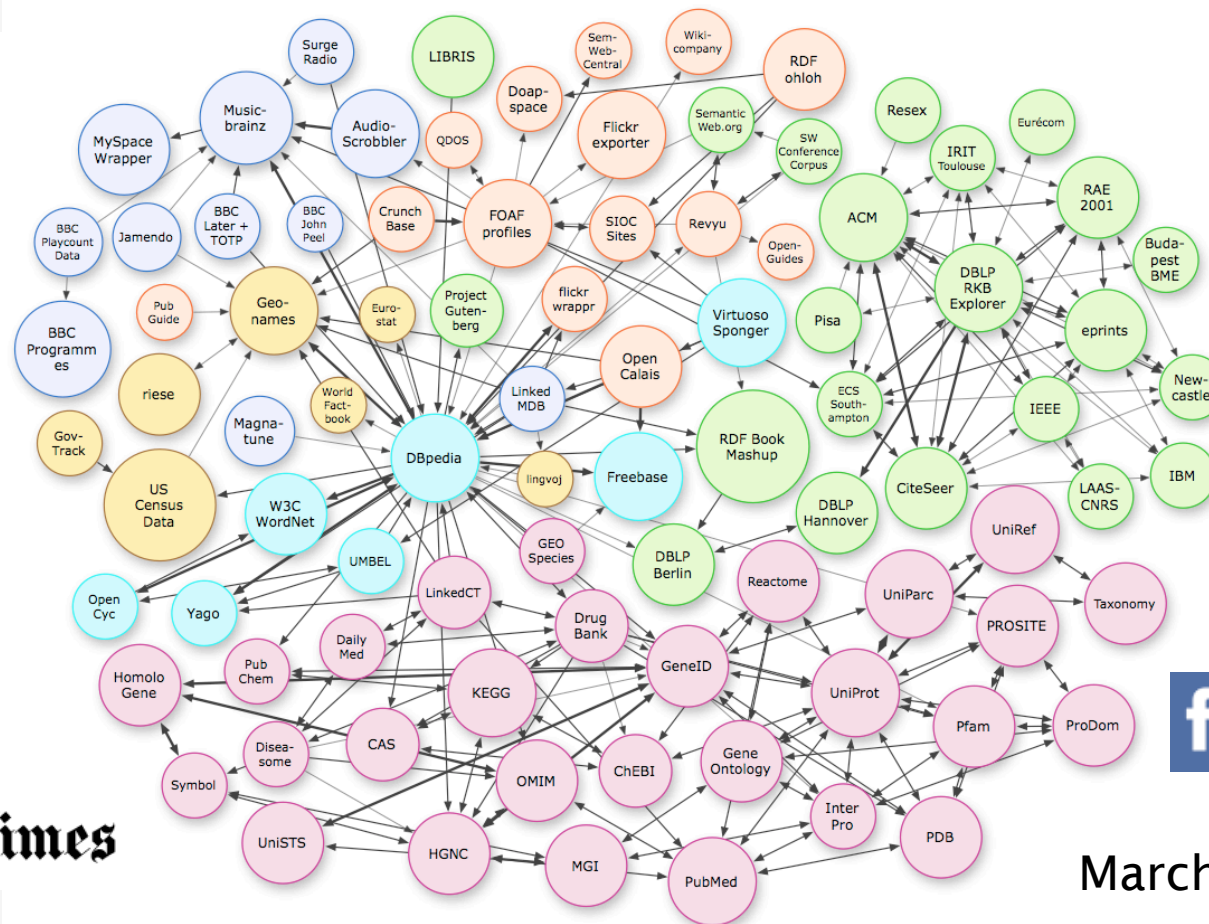
- Even if I have the FOAF data, I cannot answer the query:
  - Different identifiers used for Tim Berners-Lee
  - Who tells me that Dan Brickley is a foaf:Person?
- Need OWL and **Reasoning!**

# Linked Open Data



...

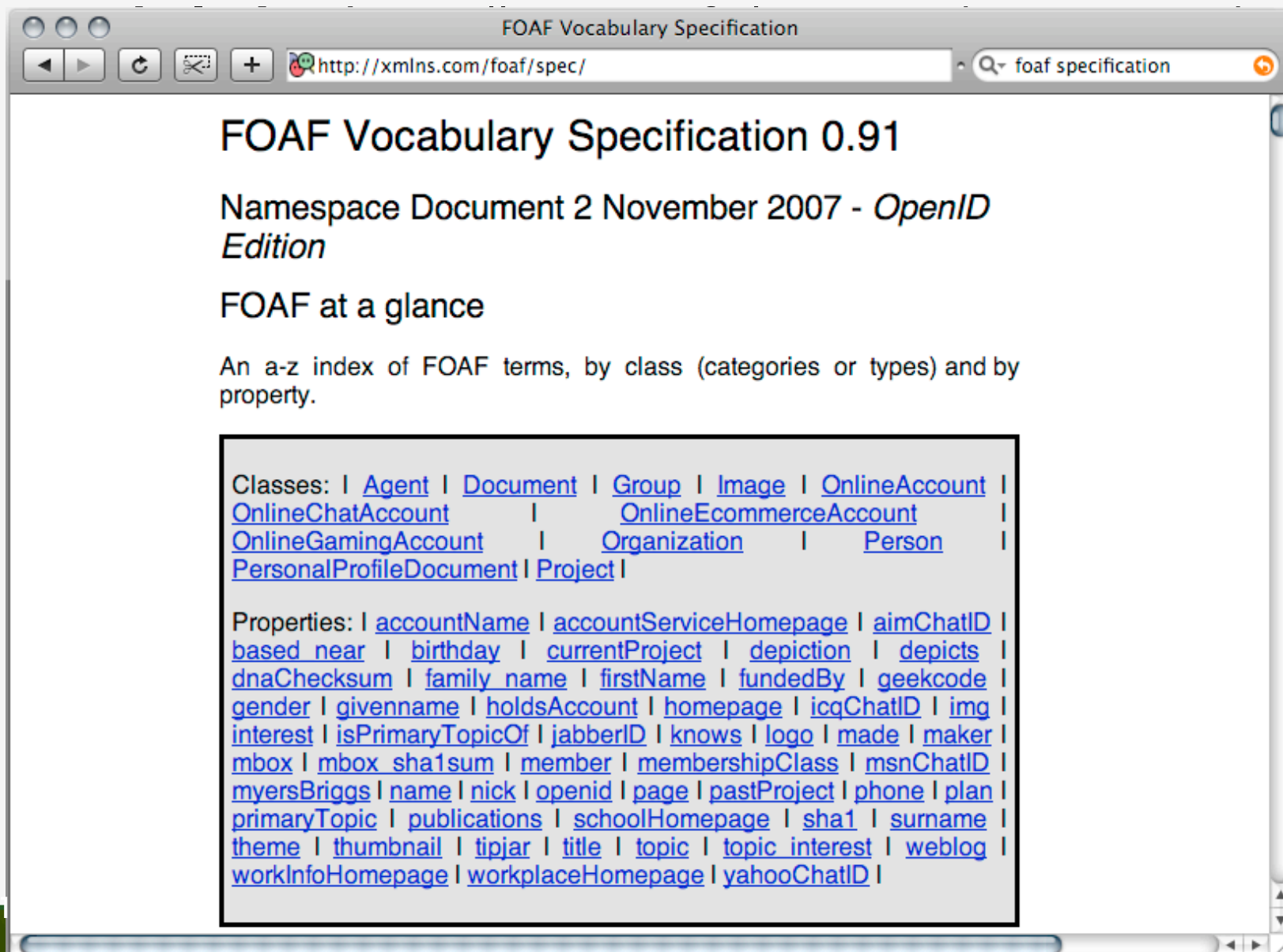
The New York Times



March 2009

Jeff Z. Pan

Excellent tutorial here: <http://www4.wiwiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/>



FOAF Vocabulary Specification

http://xmlns.com/foaf/spec/

## 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**  
*Everybody who knows someone is a Person*

$\exists \text{knows}.\top \sqsubseteq \text{Person}$

**foaf:knows rdfs:range foaf:Person**  
*Everybody who is known is a Person*

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

**foaf:Person rdfs:subclassOf foaf:Agent**  
*Everybody Person is an Agent.*

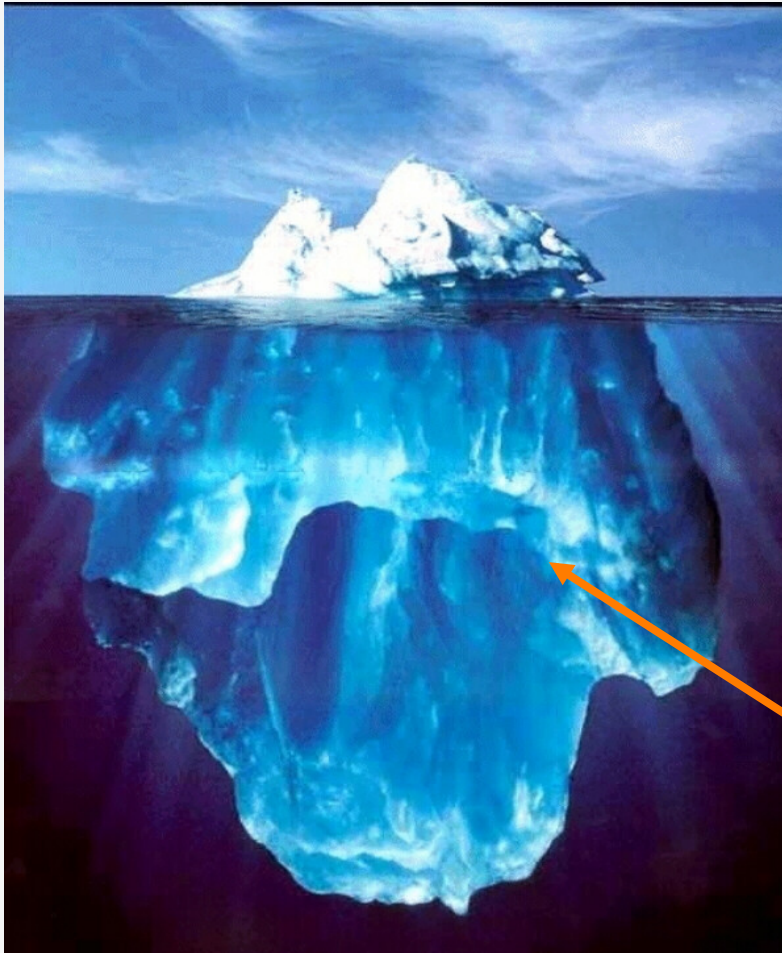
$\text{Person} \sqsubseteq \text{Agent}$

**foaf:homepage rdf:type owl:inverseFunctionalProperty .**  
*A homepage uniquely identifies its owner ("key" property)*

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

...

# Why Reasoning



Linked Data

*explicit data*

.....

*implicit data*

**Because, e.g., we  
want to exploit implicit  
data for “query  
answering”**

## Example: Why Reasoning

Query:  
Give me all buses for which Aberdeen Station is  
the origin station

```
dbp:Aberdeen_station ex:originOf ?bus.
```

```
dbp:Aberdeen_station ex:originOf nr:Line55.
```

EXPLICIT



IMPLICIT

## Example: Why Reasoning

Query:

Give me all buses for which Aberdeen Station is  
the origin station

```
dbp:Aberdeen_station ex:originOf ?bus.
```

EXPLICIT

```
dbp:Aberdeen_station ex:originOf nr:Line55.  
txc:Line1 txc:origin dgs:ABD  
dbp:Aberdeen_station owl:sameAs dgs:ABD
```

IMPLICIT



## Example: Why Reasoning

Query:

Give me all buses for which Aberdeen Station is  
the origin station

```
dbp:Aberdeen_station ex:originOf ?bus.
```

EXPLICIT



IMPLICIT

```
dbp:Aberdeen_station ex:originOf nr:Line55.  
txc:Line1 txc:origin dgs:ABD  
dbp:Aberdeen_station owl:sameAs dgs:ABD  
ex:originOf owl:inverseOf txc:origin
```

## Example: Why Reasoning

Query:  
Give me all buses for which Aberdeen Station is  
the origin station

```
dbp:Aberdeen_station ex:originOf ?bus.
```

EXPLICIT



IMPLICIT

```
dbp:Aberdeen_station ex:originOf nr:Line55.
```

```
txc:Line1 txc:origin dgs:ABD
```

```
dbp:Aberdeen_station owl:sameAs dgs:ABD
```

```
ex:originOf owl:inverseOf txc:origin
```

```
dbp:Aberdeen_station ex:originOf txc:Line1
```

# OWL2

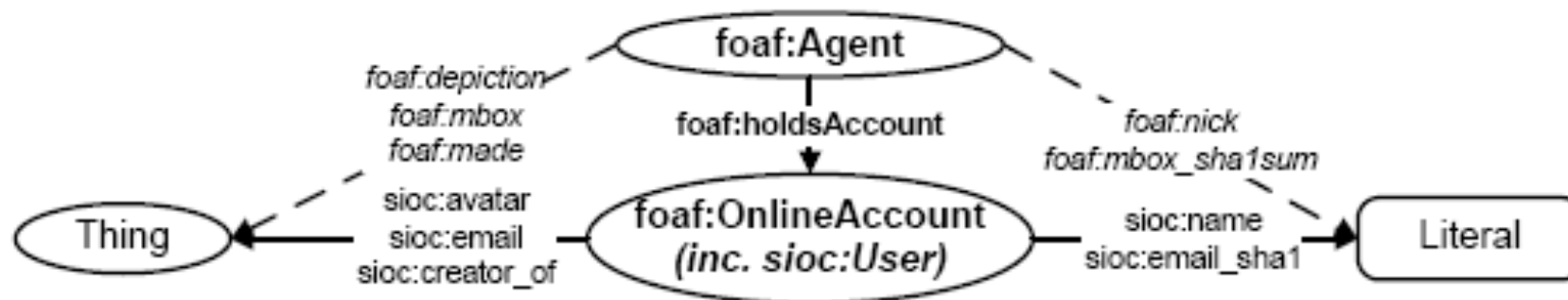
HOT!  
NEW!

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)`**



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) .
```

- *OWL DL does not allow Datatype Properties to be IFPs*
  - E.g. `foaf:mbox_sha1sum`
  - IFPs could partly be simulated using OWL2's `owl:hasKey`, as follows:  
**`owl:Thing owl:hasKey ( foaf:mbox_sha1sum ) .`**
  - means that if *\*two named individuals\** have the same `mbox_sha1sum`, then they are the same.

# 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 2006; 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

- New expressive power

- user defined datatypes, e.g.:

```
ex: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.:

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

## ■ New expressive power on properties

- qualified cardinality restrictions, e.g.:

```
_:x rdf:type owl:Restriction
_:x owl:onProperty foaf:know
_:x owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger
_:x owl:onClass Scottish
```

- 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
sioc:has_parent rdf:type owl:IrreflexiveProperty
```

- disjoint properties, e.g.:

```
sioc:parent_of owl:propertyDisjointWith sioc:has_parent
```

- keys, e.g.:

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

- **Syntactic sugar** (make things easier to say)

- Disjoint classes, e.g.:

```
_:x rdf:type owl:AllDisjointClasses
```

```
_:x owl:members (boy girl)
```

- Disjoint unions, e.g.:

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

- Negative assertions, e.g.:

```
_:x rdf:type owl:NegativePropertyAssertion
```

```
_:x owl:sourceIndividual John
```

```
_:x owl:assertionProperty foaf:know
```

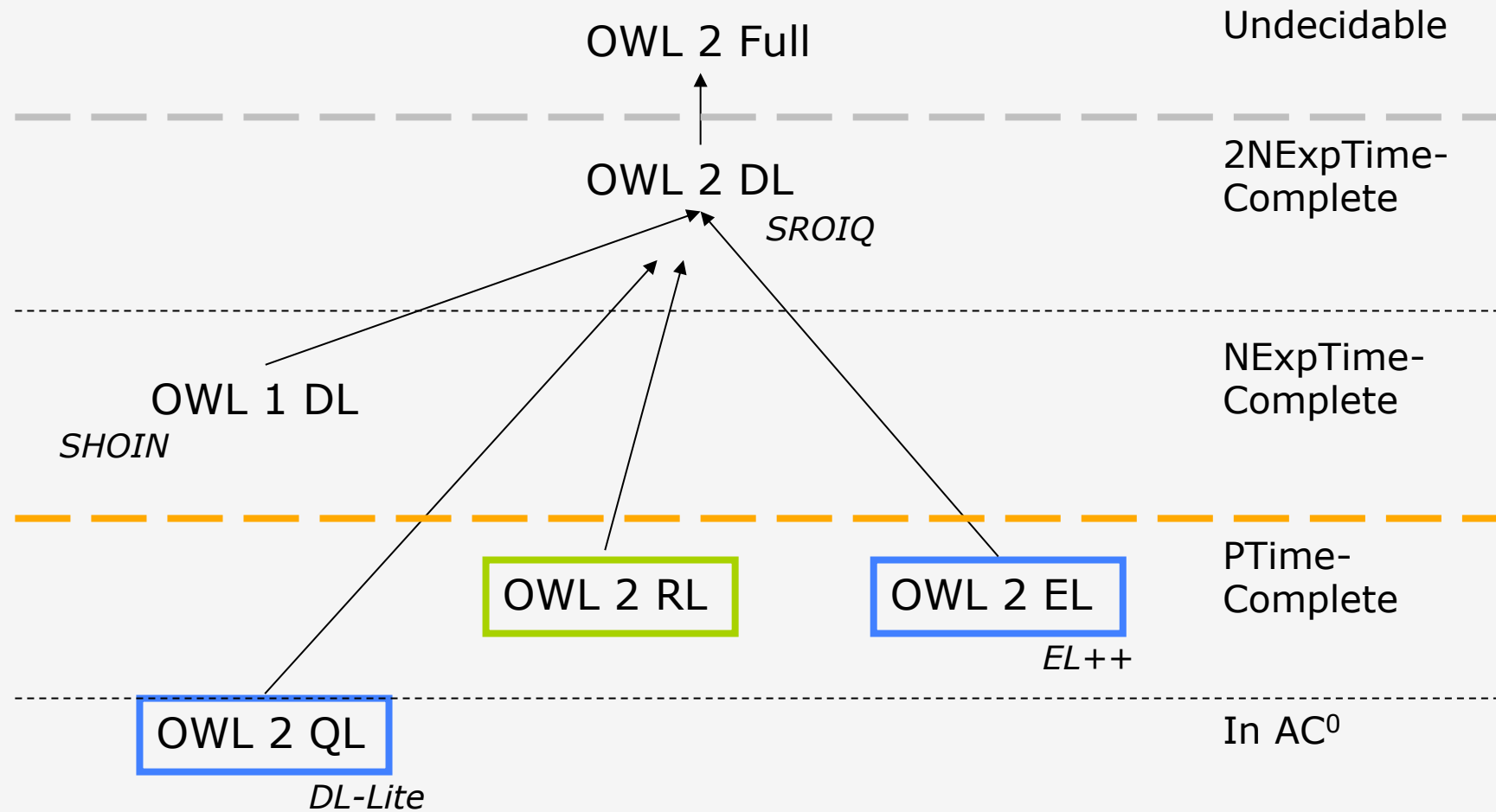
```
_:x owl:targetIndividual Mary
```

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

# OWL 2 Profiles and Reasoning Services

- Rationale:
  - Tractable
  - Tailored to specific reasoning services
- Popular reasoning services
  - TBox reasoning: OWL 2 EL
  - ABox reasoning with rule engines: OWL 2 RL
  - Query answering: OWL 2 QL
- Specification: <http://www.w3.org/TR/2009/CR-owl2-profiles-20090611/>

# The family tree



- 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 **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:

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

- Ext  
rules

# Saturation-based Technique (basics)



## Example:

ex:Performer  $\equiv$  foaf:Person  $\sqcap$   $\exists$ mo:porform\_in.mo:Performance  
mo:MusicArtist  $\equiv$  foaf:Person  $\sqcap$   $\exists$ mo:porform\_in.ex:MusicPerformance  
ex:MusicPerformance  $\sqsubseteq$  mo:Performance

ex:Performer  $\sqsubseteq$  foaf:Person  
ex:Performer  $\sqsubseteq$   $\exists$ mo:porform\_in.mo:Performance  
 $\exists$ mo:porform\_in.mo:Performance  $\sqsubseteq$  PP  
foaf:Person  $\sqcap$  PP  $\sqsubseteq$  ex:Performer

mo:MusicArtist  $\sqsubseteq$  foaf:Person  
mo:MusicArtist  $\sqsubseteq$   $\exists$ mo:porform\_in.ex:MusicPerformance  
 $\exists$ mo:porform\_in.ex:MusicPerformance  $\sqsubseteq$  PMP  
foaf:Person  $\sqcap$  PMP  $\sqsubseteq$  mo:MusicArtist

ex:MusicPerformance  $\sqsubseteq$  mo:Performance

# Saturation-based Technique (basics)



## Example:

ex:Performer  $\sqsubseteq$  foaf:Person  
ex:Performer  $\sqsubseteq \exists \text{mo:perform\_in}.\text{mo:Performance}$   
 **$\exists \text{mo:perform\_in}.\text{mo:Performance} \sqsubseteq \text{PP}$**   
foaf:Person  $\sqcap \text{PP} \sqsubseteq$  ex:Performer

mo:MusicArtist  $\sqsubseteq$  foaf:Person  
**mo:MusicArtist  $\sqsubseteq \exists \text{mo:perform\_in}.\text{ex:MusicPerformance}$**   
 $\exists \text{mo:perform\_in}.\text{ex:MusicPerformance} \sqsubseteq \text{PMP}$   
foaf:Person  $\sqcap \text{PMP} \sqsubseteq$  mo:MusicArtist

**ex:MusicPerformance  $\sqsubseteq$  mo:Performance**

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

**mo:MusicArtist  $\sqsubseteq$  PP**

# Saturation-based Technique (basics)



Example:

ex:Performer  $\sqsubseteq$  foaf:Person  
ex:Performer  $\sqsubseteq \exists$ mo:perform\_in.mo:Performance  
 $\exists$ mo:perform\_in.mo:Performance  $\sqsubseteq$  PP  
**foaf:Person  $\sqcap$  PP  $\sqsubseteq$  ex:Performer**

**mo:MusicArtist  $\sqsubseteq$  foaf:Person**  
mo:MusicArtist  $\sqsubseteq \exists$ mo:perform\_in.ex:MusicPerformance  
 $\exists$ mo:perform\_in.ex:MusicPerformance  $\sqsubseteq$  PMP  
foaf:Person  $\sqcap$  PMP  $\sqsubseteq$  mo:MusicArtist

ex:MusicPerformance  $\sqsubseteq$  mo:Performance

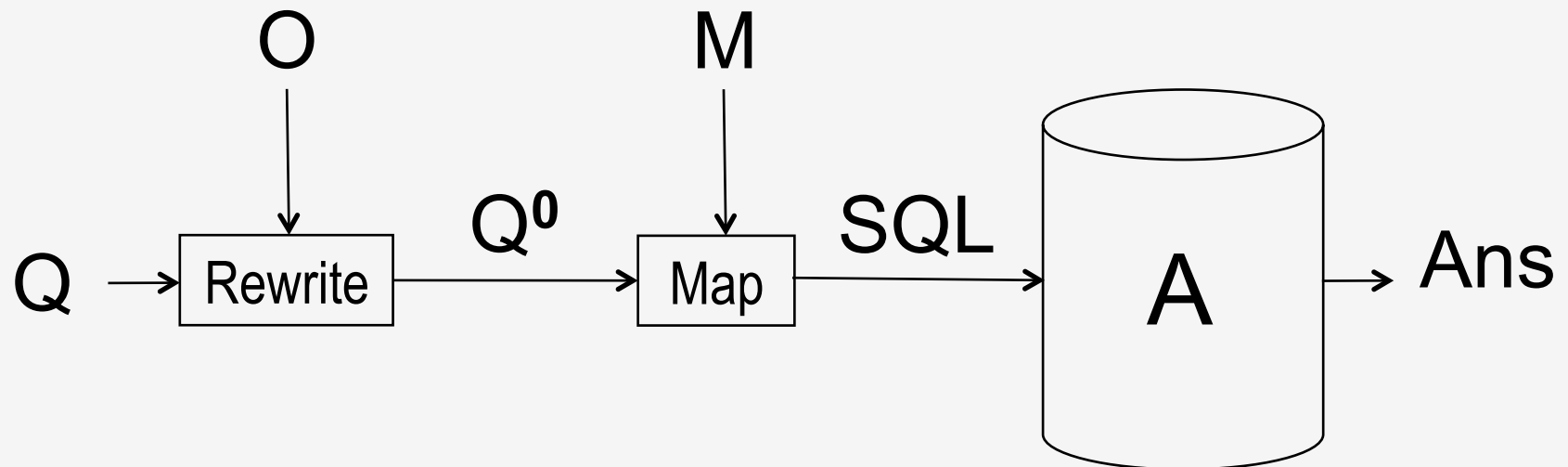
$$\frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D}{A \sqsubseteq D}$$

**mo:MusicArtist  $\sqsubseteq$  PP**  
**mo:MusicArtist  $\sqsubseteq$  ex:Performer**

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in **AC<sup>0</sup>**
- 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<sup>0</sup>) by delegating query answering to a Datalog engine [Perez-Urbina et al. 2009]

## Query Rewriting Technique (basics)

- Given ontology  $O$  and query  $Q$ , use  $O$  to rewrite  $Q$  as  $Q^0$  s.t., for any set of ground facts  $A$ :
  - $\text{ans}(Q, O, A) = \text{ans}(Q^0, ;, A)$
- Use (GAV) mapping  $M$  to map  $Q^0$  to SQL query



## Query Rewriting Technique (basics)



- Given ontology  $O$  and query  $Q$ , use  $O$  to rewrite  $Q$  as  $Q^0$  s.t., for any set of ground facts  $A$ :
  - $\text{ans}(Q, O, A) = \text{ans}(Q^0, ;, A)$
- Use (GAV) mapping  $M$  to map  $Q^0$  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:

$\text{ex:Performer} \sqsubseteq \exists \text{mo:porform\_in}.\text{mo:Performance}$

$\text{mo:MusicArtist} \sqsubseteq \text{ex:Performer}$

$\text{mo:porform\_in}(x, f(x)) \leftarrow \text{ex:Performer}(x)$

$\text{mo:Performance}(f(x)) \leftarrow \text{ex:Performer}(x)$

$\text{ex:Performer}(x) \leftarrow \text{ex:MusicArtist}(x)$

$Q(x) \leftarrow \text{mo:porform\_in}(x, y) \wedge \text{mo:Performance}(y)$

## Query Rewriting Technique (basics)

- Example:

$\text{mo:perform\_in}(x, f(x)) \leftarrow \text{ex:Performer}(x)$

$\text{mo:Performance}(f(x)) \leftarrow \text{ex:Performer}(x)$

$\text{ex:Performer}(x) \leftarrow \text{ex:MusicArtist}(x)$

$Q(x) \leftarrow \text{mo:perform\_in}(x, y) \wedge \text{mo:Performance}(y)$

$Q(x) \leftarrow \text{ex:Performer}(x) \wedge \text{mo:Performance}(f(x))$

$Q(x) \leftarrow \text{mo:perform\_in}(x, f(x)) \wedge \text{ex:Performer}(x)$

# Query Rewriting Technique (basics)



- Example:

$\text{mo:porform\_in}(x, f(x)) \leftarrow \text{ex:Performer}(x)$   
 $\text{mo:Performance}(f(x)) \leftarrow \text{ex:Performer}(x)$   
 **$\text{ex:Performer}(x) \leftarrow \text{ex:MusicArtist}(x)$**

$Q(x) \leftarrow \text{mo:porform\_in}(x, y) \wedge \text{mo:Performance}(y)$   
 $Q(x) \leftarrow \text{ex:Performer}(x) \wedge \text{mo:Performance}(f(x))$   
 $Q(x) \leftarrow \text{mo:porform\_in}(x, f(x)) \wedge \text{ex:Performer}(x)$   
 **$Q(x) \leftarrow \text{ex:Performer}(x)$**   
 **$Q(x) \leftarrow \text{ex:MusicArtist}(x)$**

- Example:

```
mo:perform_in(x, f(x)) ← ex:Performer(x)
mo:Performance(f(x)) ← ex:Performer(x)
ex:Performer(x) ← ex:MusicArtist (x)
```

```
Q(x) ← mo:perform_in(x, y) ∧ mo:Performance(y)
Q(x) ← ex:Performer (x) ∧ mo:Performance(f(x))
Q(x) ← mo:perform_in(x, f(x)) ∧ ex:Performer(x)
Q(x) ← ex:Performer(x)
Q(x) ← ex:MusicArtist (x)
```

- For OWL2 QL, result is a union of conjunctive queries

```
Q(x) ← (mo:perform_in(x, y) ∧ mo:Performance(y)) ∨ ex:Performer(x) ∨ ex:MusicArtist (x)
```

# Query Rewriting Technique (basics)



- Data can be stored/left in **RDBMS**
- Relationship between ontology and DB defined by **mappings**, e.g.:

```
ex:Performer ↔ SELECT NAME FROM PERFORMER
ex:MusicArtist ↔ SELECT NAME FROM MUSICARTIST
mo:porform_in ↔ SELECT P1NAME AND P2NAME FROM PERFORM
ex:Performance ↔ SELECT PNAME FROM PERFORMANCE
```

- UCQ translated into **SQL query**:

$$Q(x) \leftarrow (\text{mo:porform\_in}(x,y) \wedge \text{mo:Performance}(y)) \vee \text{ex:Performer}(x) \vee \text{ex:MusicArtist}(x)$$

}

```
SELECT NAME FROM PERFORMER UNION
SELECT NAME FROM MUSICARTIST UNION
SELECT PNAME FROM PERFORM, PERFORMANCE WHERE P2NAME=PNAME
```



# Stream Reasoning For Linked Data

Jeff Z. Pan and Emanuele Della Valle

