Stream Reasoning For Linked Data

Jeff Z. Pan and Emanuele Della Valle

The Web map 2008 © Tim Berners-Lee

http://www.w3.org/2007/09/map/main.jpg
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)
What is RDF

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

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

  <rdf:Description rdf:about="http://www.w3.org/People/Berners-Lee/card#i">
    <foaf:knows rdf:resource="http://dblp.l3s.de/d2r/.../Dan_Brickley">
  </rdf:Description>
  ```

RDF on the Web

- (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)
RDF on the Web

- (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/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.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 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.l3s.de/d2r/.../Berners-LeeCKSH08”

SELECT ?Name WHERE
    ?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"

```
SELECT ?Name WHERE {
{  ?CoAuthor foaf:name ?Name . }
UNION
{ ?CoAuthor foaf:knows ?Person.
?Person foaf:name ?Name }
}
```

§ Doesn't work…

§ Needs Linked Data! E.g. TimBL's FOAF file!

Back to the Data:

- DBLP:
  <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/…/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!
Jeff Z. Pan

Linked Open Data

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

Reasoning on Semantic Web Data

FOAF Vocabulary Specification

FOAF at a glance

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

- `foaf:knows rdfs:domain foaf:Person`
  - Everybody who knows someone is a Person

- `foaf:knows rdfs:range foaf:Person`
  - Everybody who is known is a Person

- `foaf:Person rdfs:subclassOf foaf:Agent`
  - Everybody Person is an Agent.

  - A homepage uniquely identifies its owner ("key" property)

- `T ≤ 1homepage`...

**Why Reasoning**

- **explicit data**
- **implicit data**

*Linked 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
\[ \text{dbp:Aberdeen_station} \text{ ex:originOf} \ ?\text{bus}. \]

\[ \text{dbp:Aberdeen_station ex:originOf nr:Line55}. \]

Example: Why Reasoning

Query:
Give me all buses for which Aberdeen Station is
the origin station
\[ \text{dbp:Aberdeen_station} \text{ ex:originOf} \ ?\text{bus}. \]

\[ \text{dbp:Aberdeen_station ex:originOf nr:Line55}. \]
\[ \text{txc:Line1 txc:origin dgs:ABD} \]
\[ \text{dbp:Aberdeen_station owl:sameAs dgs:ABD} \]
Example: Why Reasoning

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

\[
\text{dbp:Aberdeen\_station ex:originOf } \text{?bus}.
\]

\[
\text{dbp:Aberdeen\_station ex:originOf nr:Line55.}
\text{txc:Line1 txc:origin dgs:ABD}
\text{dbp:Aberdeen\_station owl:sameAs dgs:ABD}
\text{ex:originOf owl:inverseOf txc:origin}
\]

Example: Why Reasoning

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

\[
\text{dbp:Aberdeen\_station ex:originOf } \text{?bus}.
\]

\[
\text{dbp:Aberdeen\_station ex:originOf nr:Line55.}
\text{txc:Line1 txc:origin dgs:ABD}
\text{dbp:Aberdeen\_station owl:sameAs dgs:ABD}
\text{ex:originOf owl:inverseOf txc:origin}
\text{dbp:Aberdeen\_station ex:originOf txc:Line1}
\]

\[
\text{dbp:Aberdeen\_station ex:originOf txc:Line1}
\]

Can be used to enrich common ontologies on the Web

... by adding 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:

\[
\text{foaf:nick} \text{ owl:propertyChainAxiom (foaf:holdsAccount sioc:name)}
\]

- 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:

\[
\text{foaf:OnlineAccount owl:hasKey}
(\text{foaf:accountName foaf:accountServiceHomepage}).
\]

- **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 between 2004 and 2009
New Expressiveness in OWL 2

- **New expressive power**
  - user defined datatypes, e.g.:
    
    \[
    \text{ex:personAge} \equiv \text{(_:x)}
    
    \text{_:x} \text{rdf:type} rdfs:Datatype
    
    \text{_:x} \text{owl:onDatatype} xsd:integer
    
    \text{_:x} \text{owl:withRestrictions} (_:y1 _:y2)
    
    \text{_:y1} \text{xsd:minInclusive} "0"^^xsd:integer
    
    \text{_:y2} \text{xsd:maxInclusive} "150"^^xsd:integer
    
  - punning (metamodeling), e.g.:
    
    \text{ex:John rdf:type ex:Father}
    
    \text{ex:Father rdf:type ex:SocialRole}

- **New expressive power on properties**
  - qualified cardinality restrictions, e.g.:
    
    \[
    \text{_:x} \text{rdf:type} \text{owl:Restriction}
    
    \text{_:x} \text{owl:onProperty} \text{foaf:know}
    
    \text{_:x} \text{owl:minQualifiedCardinality} "2"^^xsd:nonNegativeInteger
    
    \text{_:x} \text{owl:onClass Scottish}
    
  - property chain inclusion axioms, e.g.:
    
    \text{foaf:nick owl:propertyChainAxiom} (\text{foaf:holdsAccount sioc:name})
    
  - local reflexivity restrictions, e.g.:
    
    \[
    \text{_:x} \text{rdf:type} \text{owl:Restriction}
    
    \text{_:x} \text{owl:onProperty} \text{like}
    
    \text{_:x} \text{owl:hasSelf} "true"^^xsd:boolean  \text{[for narcissists]}
    
  - reflexive, irreflexive, symmetric, and antisymmetric properties, e.g.:
    
    \text{foaf:know rdf:type} \text{owl:ReflexiveProperty}
    
    \text{sioc:has_parent rdf:type} \text{owl:IrreflexiveProperty}
    
  - disjoint properties, e.g.:
    
    \text{sioc:parent_of owl:propertyDisjointWith sioc:has_parent}
    
  - keys, e.g.:
    
    \text{foaf:OnlineAccount owl:hasKey}
    
    (\text{foaf:accountName} \text{foaf:accountServiceHomepage})
New Expressiveness in OWL 2

- **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
    ```

OWL 2 DL

- 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/owl2-profiles/](http://www.w3.org/TR/owl2-profiles/)

---

The family tree

- OWL 2 Full
- OWL 2 DL
- OWL 1 DL
- OWL 2 QL
- OWL 2 EL

- Undecidable
- 2NExpTime-Complete
- NExpTime-Complete
- PTime-Complete
- In AC^0
OWL 2 EL

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime \(\text{(PTime-Complete)}\)
  - Data complexity of query answering also PTime-Complete

- Based on \textbf{EL} family of description logics [Baader et al. 2005]

- Can exploit \textit{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:
  
  \[ \begin{align*}
  A \sqsubseteq B & \quad A \sqcap B \sqsubseteq C & \quad A \sqsubseteq \exists R.B & \quad \exists R.B \sqsubseteq C \\
  A \sqsubseteq C & \quad A \sqsubseteq C & \quad A \sqsubseteq D & \quad A \sqsubseteq D
  \end{align*} \]

- Saturate using inference rules:

- Ext rules
Saturation-based Technique (basics)

Example:

```
ex:Performer ⊑ foaf:Person ⊑ mo:perform_in ⊑ mo:Performance
mo:MusicArtist ⊑ foaf:Person ⊑ mo:perform_in ⊑ mo:Performance ⊑ ex:MusicPerformance
ex:MusicPerformance ⊑ mo:Performance
```

```
ex:Performer ⊑ foaf:Person
ex:Performer ⊑ mo:perform_in ⊑ mo:Performance
mo:perform_in ⊑ mo:Performance ⊑ PP
mo:Performance ⊑ PP ⊑ ex:Performer

mo:MusicArtist ⊑ foaf:Person
mo:MusicArtist ⊑ mo:perform_in ⊑ mo:MusicPerformance
mo:perform_in ⊑ mo:MusicPerformance ⊑ PMP
mo:Performance ⊑ PMP ⊑ mo:MusicArtist

ex:MusicPerformance ⊑ mo:Performance
```

Saturation-based Technique (basics)

Example:

```
ex:Performer ⊑ foaf:Person
ex:Performer ⊑ mo:perform_in ⊑ mo:Performance
mo:perform_in ⊑ mo:Performance ⊑ PP
mo:Performance ⊑ PP ⊑ ex:Performer

mo:MusicArtist ⊑ foaf:Person
mo:MusicArtist ⊑ mo:perform_in ⊑ mo:MusicPerformance
mo:perform_in ⊑ mo:MusicPerformance ⊑ PMP
mo:Performance ⊑ PMP ⊑ mo:MusicArtist

ex:MusicPerformance ⊑ mo:Performance
```

```
A ⊑ B ⊑ C ⊑ R.C ⊑ D
A ⊑ D
mo:MusicArtist ⊑ PP
```

```
ex:Performer ⊑ foaf:Person
ex:Performer ⊑ mo:perform_in ⊑ mo:Performance
mo:perform_in ⊑ mo:Performance ⊑ PP
mo:Performance ⊑ PP ⊑ ex:Performer

mo:MusicArtist ⊑ foaf:Person
mo:MusicArtist ⊑ mo:perform_in ⊑ mo:MusicPerformance
mo:perform_in ⊑ mo:MusicPerformance ⊑ PMP
mo:Performance ⊑ PMP ⊑ mo:MusicArtist

ex:MusicPerformance ⊑ mo:Performance
```

Saturation-based Technique (basics)
Saturation-based Technique (basics)

Example:

A ⊆ B  A ⊆ C  B ∩ C ⊆ D
d

OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in AC⁰
- 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 AC⁰) 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

Resolution based query rewriting
- \textbf{Clausify} ontology axioms
- \textbf{Saturate} (clausified) ontology and query using resolution
- \textbf{Prune} redundant query clauses
Query Rewriting Technique (basics)

- Example:

\[
\text{ex:Performer} \subseteq \text{mo:perform.in, mo:Performance}
\]
\[
\text{mo:MusicArtist} \subseteq \text{ex:Performer}
\]
\[
\text{mo:perform.in}(x, f(x)) \rightarrow \text{ex:Performer}(x)
\]
\[
\text{mo:Performance}(f(x)) \rightarrow \text{ex:Performer}(x)
\]
\[
\text{ex:Performer}(x) \rightarrow \text{ex:MusicArtist}(x)
\]

\[
Q(x) \leftarrow \text{mo:perform.in}(x, y) \land \text{mo:Performance}(y)
\]

\[
Q(x) \leftarrow \text{ex:Performer}(x) \land \text{mo:Performance}(f(y))
\]

\[
Q(x) \leftarrow \text{mo:perform.in}(x, f(x)) \land \text{ex:Performer}(x)
\]
Example:

\[
\text{Q}(x) \leftarrow \text{mo:perform}_\text{in}(x,y) \land \text{mo:Performance}(y)
\]

\[
\text{Q}(x) \leftarrow \text{ex:Performer}(x) \land \text{mo:Performance}(f(x))
\]

\[
\text{Q}(x) \leftarrow \text{ex:Performer}(x) \land \text{mo:perform}_\text{in}(x,f(x))
\]

\[
\text{Q}(x) \leftarrow \text{ex:MusicArtist}(x)
\]

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

\[
\text{Q}(x) \leftarrow (\text{mo:perform}_\text{in}(x,y) \land \text{mo:Performance}(y) \lor \text{ex:Performer}(x) \lor \text{ex:MusicArtist}(x)
\]
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:perform_in ↔ SELECT P1NAME AND P2NAME FROM PERFORM
  ex:Performance ↔ SELECT PNAME FROM PERFORMANCE
  ```

- UCQ translated into SQL query:

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