Stream Reasoning For Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell’Aglio, E. Della Valle, and J.Z. Pan
http://streamreasoning.org/sr4ld2013

OWL Reasoning and Stream Reasoning with LOD
Jeff Z. Pan
University of Aberdeen
http://homepages.abdn.ac.uk/jeff.z.pan/pages/
Share, Remix, Reuse — Legally

- This work is licensed under the Creative Commons Attribution 3.0 Unported License.

- **Your are free:**
  - **to Share** — to copy, distribute and transmit the work
  - **to Remix** — to adapt the work

- **Under the following conditions**
  - **Attribution** — You must attribute the work by inserting
    - “[source http://streamreasoning.org/sr4ld2013](http://streamreasoning.org/sr4ld2013)” at the end of each reused slide
    - a credits slide stating

- To view a copy of this license, visit [http://creativecommons.org/licenses/by/3.0/](http://creativecommons.org/licenses/by/3.0/)
LOD = Linked Open Data / Linked Ontological Data
Agenda

1. OWL Reasoning with LOD (30m)
2. OWL Stream Reasoning with LOD (40m)
3. Hands-on session (20m)
What is an Ontology

A formal dictionary of domain vocabulary
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Koala
- Specifies meaning (semantics) of terms
  - Koala eat only some part of Eucalypt
  - Koala $\subseteq \forall \text{eat.}(\exists \text{part of Eucalypt})$
- Eucalypt is Plant
  - $\text{Eucalypt} \subseteq \text{Plant}$
Components of Ontology

- A TBox (Terminology Box) is a set of “schema” axioms (sentences), e.g.:
  
  \[
  \begin{align*}
  \text{Koala} & \subseteq \forall \text{eat.}(\exists \text{partof.Eucalypt}) \\
  \text{Eucalypt} & \subseteq \text{Plant}
  \end{align*}
  \]

  - i.e., a **background theory for the vocabulary**

- An ABox (Assertion Box) is a set of “data” axioms (ground facts), e.g.:
  
  gummy: Koala
Infer **implicit** knowledge from **explicit** knowledge

\[ Koala \sqsubseteq \forall eat. (\exists \text{partof}. Eucalypt) \]
\[ Eucalypt \sqsubseteq \text{Plant} \]

\[ Koala \sqsubseteq \forall eat. \exists \text{partof}. \text{Plant} \]
\[ \text{Plant} \sqsubseteq \exists \text{partof}. \text{Plant} \sqsubseteq \text{VegeFood} \]

\[ Koala \sqsubseteq \forall eat. \text{VegeFood} \]
\[ \forall eat. \text{VegeFood} \sqsubseteq \text{Herbivore} \]

\[ Koala \sqsubseteq \text{Herbivore} \]
Example:

```
SELECT ?X, ?Y
FROM <http://example.org/animal.owl>
WHERE {?X eat ?Y .}
```

```
SELECT ?X
FROM <http://example.org/animal.owl>
WHERE {?X rdf:type Herbivore .}
```
The need for bridging the gap between data and queries

- Question Answering
- Semantic Search
- Social Discovery
- Informed Rural Passenger
- Digital Economy

http://streamreasoning.org/sr4ld2013
• 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:
  - $A \subseteq B \quad A \cap B \subseteq C \quad A \subseteq \exists R.B \quad \exists R.B \subseteq C$

- Saturate using inference rules:
  - $A \subseteq B \quad B \subseteq C$
  - $A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D$
  - $A \subseteq C$
  - $A \subseteq D$

- Extension to Horn fragment requires (many) more rules
Example:

```turtle
ex:Performer = foaf:Person \ nho:mo:perform_in.mo:Performance
mo:MusicArtist = foaf:Person \ nho:mo:perform_in. ex:MusicPerformance
ex:MusicPerformance c mo:Performance

ex:Performer c foaf:Person
ex:Performer c nho:mo:perform_in.mo:Performance
nho:mo:perform_in.mo:Performance c PP
foaf:Person n PP c ex:Performer

mo:MusicArtist c foaf:Person
mo:MusicArtist c nho:mo:perform_in.ex:MusicPerformance
nho:mo:perform_in.ex:MusicPerformance c PMP
foaf:Person n PMP c mo:MusicArtist

ex:MusicPerformance c mo:Performance
```
Saturation-based Technique (basics)

Example:

- ex:Performer ∈ foaf:Person
- ex:Performer ∈ mo:perform_in.mo:Performance
- mo:perform_in.mo:Performance ⊑ PP
- foaf:Person ⊓ PP ⊑ ex:Performer

- mo:MusicArtist ∈ foaf:Person
- mo:MusicArtist ∈ mo:perform_in.ex:MusicPerformance
- mo:perform_in.ex:MusicPerformance ⊑ PMP
- foaf:Person ⊓ PMP ⊑ mo:MusicArtist

- ex:MusicPerformance ⊑ mo:Performance

Diagram:

```
A ⊑ ∃R.B  B ⊑ C  ∃R.C ⊑ D  
   ⊑               ⊑              
   ⊑               ⊑                
   ⊑               ⊑                  
   ⊑               ⊑                    
   ⊑               ⊑                        
A ⊑ D

mo:MusicArtist ⊑ PP
```
Saturation-based Technique (basics)

Example:

\[ A \subseteq B \quad A \subseteq C \quad B \cap C \subseteq D \quad A \subseteq D \]

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

ex:Performer \sqsubseteq foaf:Person
ex:Performer \sqsubseteq \exists mo:perform_in.mo:Performance
\exists mo:perform_in.mo:Performance \sqsubseteq PP
foaf:Person \cap 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 \cap PMP \sqsubseteq mo:MusicArtist

ex:MusicPerformance \sqsubseteq mo:Performance
OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering in $\text{AC}^0$

- Based on **DL-Lite** family of description logics
  [Calvanese et al. 2005; 2006; 2008]

- Can exploit **query rewriting** based reasoning technique
  - 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 $\text{AC}^0$) 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

```
Q → Rewrite → Q^0 → Map → SQL → A → Ans
```
Idea: to compile a source ontology \( O \) (in more expressive \( L_S \)) into its upper/lower bound (in less expressive \( L_T \))

- Entailment set \( ES(O, L_T) \) of \( O \) in \( L_T \)
  - The set of all \( L_T \) axioms that are entailed by \( O \) under \( N_C \), \( N_P \) and \( N_I \)
Semantic Approximation [Pan and Thomas, 2007]

- Strongest weaker approximation for QL $\text{ES}(O, \text{DL-Lite}_{\text{core}})$ of an OWL2 DL $O$ is finite and unique.

- **Theorem 1**: Given an ontology $O$, a conjunctive query $q(X)$ and an evaluation $[X \rightarrow S]$, if $\text{ES}(O, \text{DL-Lite}_{\text{core}}) \models q_{[X \rightarrow S]}$, then $O \models q_{[X \rightarrow S]}$.

- **Theorem 2**: Given an ontology $O_S$, a database-style conjunctive query $q(X)$ without non-distinguished variables and an evaluation $[X \rightarrow S]$, $\text{ES}(O_S, \text{DL-Lite}_{\text{core}}) \models q_{[X \rightarrow S]}$ iff $O_S \models q_{[X \rightarrow S]}$. 

$\text{Q}$  

$\text{ES}(O, L_T)$
Syntactic Approximate Reasoning
[Ren et al, 2010]

- Syntactic approximation from OWL2 DL to OWL2 EL
  - Minor syntactic gap results in major complexity difference
  - Using approximation to bridge the gap

<table>
<thead>
<tr>
<th>DL ROQ (large subset of OWL2 DL)</th>
<th>DL EL++ (large subset of OWL2 EL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r. C \mid {a} \mid \neg C \geq n R. C$</td>
<td>$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r. C \mid {a}$</td>
</tr>
<tr>
<td>$C \sqsubseteq D$</td>
<td>$r \sqsubseteq s, r_1 \circ \ldots \circ r_n \sqsubseteq s$</td>
</tr>
<tr>
<td>$a : C'$</td>
<td>$(a, b) : r$</td>
</tr>
</tbody>
</table>

N2EXPTIME-complete                      PTIME-complete
Example: Syntactic Approximations

- Represent non-OWL2-EL concepts with fresh named concepts
  - E.g., $\forall r. C$ subClassOf D $\Rightarrow A_{\forall r. C}$ subClassOf D

- Maintain semantic relations for these named concepts
  - complementary relations
  - cardinality relations

- Additional tractable completion Rules (on top of the EL ones), e.g.
  - Handling complement
    - E.g. $B$ subClassOf $C \Rightarrow \neg C$ subClassOf $\neg B$
## Evaluation: Syntactic Approximations

### Table 15: Ontology Materialisation Comparison (t/o: out of time)

<table>
<thead>
<tr>
<th>Ontology $\mathcal{O}$</th>
<th>FaCT++</th>
<th>HermiT</th>
<th>Pellet</th>
<th>REL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semintec</td>
<td>t/o</td>
<td>4.661</td>
<td>6.172</td>
<td>2.774 100%</td>
</tr>
<tr>
<td>VICODI</td>
<td>t/o</td>
<td>18.282</td>
<td>9.828</td>
<td>3.101 100%</td>
</tr>
<tr>
<td>Wine</td>
<td>364.8</td>
<td>47.865</td>
<td>200.982</td>
<td>0.172 100%</td>
</tr>
<tr>
<td>Sweet Numerics</td>
<td>2.139</td>
<td>32.419</td>
<td>2.016</td>
<td>0.578 100%</td>
</tr>
<tr>
<td>MGED</td>
<td>t/o</td>
<td>7.247</td>
<td>449.807</td>
<td>0.234 100%</td>
</tr>
<tr>
<td>BoC small</td>
<td>484.32</td>
<td>32.045</td>
<td>t/o</td>
<td>0.578 100%</td>
</tr>
<tr>
<td>BoC middle</td>
<td>t/o</td>
<td>426.013</td>
<td>t/o</td>
<td>3.348 100%</td>
</tr>
<tr>
<td>BoC big</td>
<td>t/o</td>
<td>t/o</td>
<td>t/o</td>
<td>22.058 N/A</td>
</tr>
</tbody>
</table>

### Table 12: TBox Classification Comparison (t/o: out of time; e/o: terminate with error)

<table>
<thead>
<tr>
<th>Ontology $\mathcal{O}$</th>
<th>FaCT++</th>
<th>HermiT</th>
<th>Pellet</th>
<th>REL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biological Process</td>
<td>2.792</td>
<td>5.796</td>
<td>9.383</td>
<td>2.499 100%</td>
</tr>
<tr>
<td>Cellular Component</td>
<td>4.648</td>
<td>9.186</td>
<td>12.36</td>
<td>4.513 100%</td>
</tr>
<tr>
<td>GO</td>
<td>2.729</td>
<td>5.904</td>
<td>9.461</td>
<td>2.499 100%</td>
</tr>
<tr>
<td>Cyc</td>
<td>14.36</td>
<td>11.152</td>
<td>140.15</td>
<td>2.514 100%</td>
</tr>
<tr>
<td>FMA Constitutional</td>
<td>e/o</td>
<td>t/o</td>
<td>t/o</td>
<td>57.589 N/A</td>
</tr>
<tr>
<td>DLP</td>
<td>0.188</td>
<td>34.409</td>
<td>t/o</td>
<td>0.156 100%</td>
</tr>
<tr>
<td>EFO</td>
<td>2.825</td>
<td>3.201</td>
<td>t/o</td>
<td>0.531 100%</td>
</tr>
<tr>
<td>OBI</td>
<td>93.66</td>
<td>172.311</td>
<td>t/o</td>
<td>1.943 100%</td>
</tr>
<tr>
<td>NCIt</td>
<td>44.24</td>
<td>810.87</td>
<td>360.582</td>
<td>34.801 100%</td>
</tr>
</tbody>
</table>
Agenda

1. OWL Reasoning with LOD (30m)
2. OWL Stream Reasoning with LOD (40m)
3. Hands-on session (20m)
LOD Streams

- **Two streams [Ren and Pan, 2011]**
  - to-erase stream
  - to-add stream

- A LOD stream O[0,n] is a sequence of classical ontologies O(0), O(1), ..., O(n):
  - O(0) is the initial ontology
  - Er(i) axioms to erase from O(i)
  - Ad(i) axioms to add into O(i)
  - O(i+1) = O(i) U Ad(i) \ Er(i)

- **Key task**
  - Answer a set of monitoring queries at each snapshot ontology O(i)
The DRed (Delete and Re-derive) approach [Volz et. al. 2005]
- Maintaining the materialisation of the knowledge base
- Over-delete impacted entailments
- Re-derive impacted entailments
- Derive new entailments

Key techniques
- Delete: justification
- Re-derive: incremental reasoning
Justification: Key Enabler for Delete

- **Justification**
  - Given an ontology O and a reasoning result rs
  - A justification J(rs) is a **minimal** subset of O that imply rs
  - There could be multiple justifications

- **Challenges:**
  - Computing one justification for OWL2-DL is costly
  - Computing all justifications is NP-complete even for OWL2 tractable profiles

- **One justification at a time is needed**
  - If the current justification J(rs) and Er(i) overlap
  - then rs should be removed as well
A directed graph
- Nodes: axioms / entailments
- Edges: derivation relations among axioms / entailments
- All entailments are reachable from their justifications
  - Easy to identify impacted entailments

Truth Maintenance System

- Truth Maintenance System
- http://streamreasoning.org/sr4ld2013
**Delete and Re-derive with TMS**

- **Erasing**
  - Remove all nodes reachable from the erased axioms
  - Removing all corresponding edges

- **Adding**
  - Adding added axioms as new nodes into the graph
  - Inferring new results
  - Establishing new edges

*Diagram showing examples of interest, shareInterest, interest, SemanticWeb, Alex, David, Post, tweet, retweet, follow.*
Key Challenges

- **Q1**: How to efficiently perform reasoning and compute **justifications**?
  - Q1.1 tractable profiles such as OWL2 EL
  - Q1.2 OWL2 DL

- **Q2**: How to perform **incremental reasoning**
  - based on justifications
Normalise ontology axioms to standard form:

\[ A \subseteq B \quad A \cap B \subseteq C \quad A \subseteq \exists R.B \quad \exists R.B \subseteq C \]

Saturate using inference rules:

\[ \begin{align*}
A \subseteq B & \quad B \subseteq C & \quad A \subseteq B & \quad A \subseteq C & \quad B \cap C \subseteq D \\
A \subseteq C & \quad A \subseteq D & \quad A \subseteq \exists R.B & \quad B \subseteq C & \quad \exists R.C \subseteq D \\
\end{align*} \]

Applicable approaches

- data driven forward chaining
- query driven forward chaining
Stream Reasoning in OWL 2 EL

[Ren and Pan, 2011]

- Optimised memory consumption
  - **Reduce** the number of maintained nodes and edges

- We entail an axiom \( C \sqsubseteq \exists r.D \) if it is **classification-relevant**, i.e. contributing to the reasoning results we are looking for:
  - E.g. there is some \( \exists r.A \sqsubseteq B \)
  - or, \( r \sqsubseteq s \) and \( s \) is classification-relevant
  - or, ...

\[ \text{ActiveTalk} \sqsubseteq \exists \text{in.Session} \]
\[ \{\text{talk1}\} \sqsubseteq \text{in.Session} \]
\[ \{\text{talk1}\} \sqsubseteq \text{ActiveTalk} \]

*in* is **NOT** classification-relevant

We do **NOT** need to compute or maintain the result
- Reasoning in OWL 2 QL is based on query rewriting
- Given ontology $O$ and query $Q$, use $O$ to rewrite $Q$ as $Q^0$ s.t., for any set of ground facts $A$:
  - $\text{ans}(Q, O, A) = \text{ans}(Q^0, ;, A)$
- Some related new results:
  - Incremental query rewriting [Venetis et. al. 2012]
  - Temporal DL-Lite [Borgwardt et. al., 2013]
  - DL-Lite with aggregate queries [Kostylev and Reutter, 2013]
- Still be room for further research
Stream Reasoning in OWL 2 DL

- Generate a TMS when doing approximation and reasoning
  - Nodes:
    - Asserted axioms;
    - Approximated axioms;
    - Entailed axioms;
  - Edges:
    - Created during approximation and reasoning

```
interest ◦ interest ⊑ shareInterest
(David, SemanticWeb) : interest
(Alex, SemanticWeb) : interest

{Alex} ⊑ ∃interest.{SemanticWeb}
{SemanticWeb} ⊑ ∃invInterest.{David}
interest ◦ invInterest ⊑ shareInterest

{Alex} ⊑ ∃shareInterest.{David}
```
Stream Reasoning with Syntactic Approximated-TMS

Original Ontology:
- interest o interest - □ shareInterest
- (David, SemanticWeb) : interest
- (Alex, SemanticWeb) : interest
- Alex ≠ David
- (Alex, Post3) : retweet
- retweet o tweet - □ follow

Approximated Ontology:
- \{Alex\} ⊨ ∃ interest . \{SemanticWeb\}
- \{SemanticWeb\} ⊨ ∃ invInterest . \{David\}
- interest o invInterest - □ shareInterest
- \{Alex\} ⊨ nDavid
- retweet o invTweet - □ follow
- \{Alex\} ⊨ ∃ retweet . \{Post3\}
- \{Post3\} ⊨ ∃ invTweet . \{David\}

Approximated Question:
- nDavid \□ ∃ shareInterest . \{David\} \□ ∃ follow . \{David\}

Original Query:
- \{David\} \□ ∃ shareInterest . \{David\} \□ ∃ follow . \{David\}
Stream Reasoning with Syntactic Approximated-TMS

- **Improvement:**
  - The approximation can be reused when adding new axioms
  - Reasoning complexity is reduced

- **Quality:**
  - Soundness-guaranteed
    - Entailed axioms will never be over-looked in erasing
Evaluation

- LUBM
  - Arbitrarily large ABox
  - Non-trivial reasoning
    - Different possible sources of a same entailment

- Stream simulation
  - Generating a LUBM ontology
  - Preserve the TBox through the stream
    - To ensure the difficulty of reasoning
  - ABox stream
    - Partitioning the ABox
    - Swap sub-ABoxes in stream updating
Evaluation

- Criteria
  - Absolute volume of the stream
  - Relative volume of the update
  - Absolute time of updating
  - Relative time of updating

![Graph showing the evaluation criteria and results](http://streamreasoning.org/sr4ld2013)
A brief summary

- **Advantage**
  - Significantly **reduce** the needed number of entailment checking

- **Disadvantages & optimisations to address them**
  - **Require additional computation** to form the edges
    - Using saturation-based algorithms to generate the TMS on-the-fly
      - EL+, Horn-SHIQ vs. Tableau-based or Datalog-based algorithms
  - **Significantly increase the memory consumption**
    - Saturation algorithms already make intermediate results more “reusable”
    - Optimising the algorithm to further minimise intermediate results
  - **High complexity in expressive languages**
    - approximation

Combining stream reasoning with local closed world reasoning

- Useful in deployment lifecycle
  - deployed components should be closed (in principle)
  - undeployed components remain open

- Non-monotonic reasoning with stream
Local closed world reasoning in NBox

Motivation:
- There are knowledge and data that users have complete knowledge about
- such as spicy dishes

Solution: NBox (Negation as failure Box) [JTST2010]
- TBox: a set of schema axioms
- ABox: a set of data axioms
- NBox: a set of closed concepts and roles
  - Declarative: Permanently closed, with annotation property
  - Runtime: temporally closed, with API provided

<table>
<thead>
<tr>
<th>Food</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curry Chicken</td>
<td>Minor Spicy</td>
</tr>
<tr>
<td>Salmon Fillet</td>
<td></td>
</tr>
<tr>
<td>Spicy Grilled Shrimp</td>
<td>Spicy</td>
</tr>
<tr>
<td>Pepper Salad</td>
<td>Vege</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Vegetarian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>No</td>
</tr>
<tr>
<td>Yuting</td>
<td>Yes</td>
</tr>
<tr>
<td>Jek</td>
<td></td>
</tr>
<tr>
<td>Yuan</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: NBox Reasoning

- Local closed world reasoning in DMIL

Data source port type:
- Query 1 data source
- Classifier data
- Merge data 1

SQL data port type:
- Query 1 SQL data

Connection:
- SQL connection 1
- Class connection

Input Port:
- ONLY
- Some connection

Output Port:
- ONLY

Which input port can be used here?
Example: NBox Reasoning

- Local closed world reasoning in DMIL

```
Data source port type

Query 1 data source
Classifier data
Merge data 1

Input Port

ONLY
Some connection
ONLY

Connection

SQL connection 1
class connection

Output Port

SQL data port type

Query 1 SQL data

Which input port can NOT be used here?

Example: NBox Reasoning
```
Local closed world reasoning in DMIL

- Data source port type
  - Restricted connection
  - Occupied
  - Merge data 1

- SQL data port type
  - Query 1 SQL data

- Connection
  - SQL connection 1
  - class connection

Which input port can NOT be used here?

Example: NBox Reasoning
Example: NBox Reasoning

- Local closed world reasoning in DMIL

Data source port type

- Restricted connection
- Occupied
- Valid input port

Only
- Some connection

SQL data port type

- Query 1 SQL data

Which input port can be used here?

Connection

Input Port

Output Port

SQL connection 1

class connection

Valid input port

Occupied

Restricted connection

Input Port

Output Port

Only

Some connection

SQL data

port type

port type

port type

port type

http://streamreasoning.org/sr4ld2013

http://most-project.eu
A brief note on NBox

- Adding NBox into OWL DL/OWL 2 DL won’t increase the complexity for reasoning
  - some approximate reasoning algorithm is available in implemented in TrOWL [Ren et. al. 2010]

- However, adding NBox would make DL-Lite and EL intractable in general
  - unless some safe conditions are satisfied [Lutz et. al. 2012]
Research Challenges

- Dealing with noisy and uncertain data
- Stream LOD reasoning in other OWL2 profiles
  - Such as QL and RL
- Further optimisations techniques
  - Such as decomposition and parallelisation
- Other notions of stream reasoning
- Relations between stream reasoning and temporal reasoning
Approximate reasoning and TMS play key roles in OWL 2 DL stream reasoning
Where to find more information

TrOWL (with publication list and tutorials):

http://trowl.eu/

http://trowl.eu/owl-dbc/
1. OWL Reasoning with LOD (30m)
2. OWL Stream Reasoning with LOD (40m)
3. Hands-on session (20m)
Researchers present their works in conference sessions.

- “Introducing HermiT2: an NLogSpace Reasoner of OWL2” – by Ian Horrocks
- @ Session of Reasoning

Researchers activities can be collected from mobile sensors (e.g. RFID) or web crawlers.

- @Jeff: “Ian just gave an very interesting presentation about his new paper”
- @Yuan: “I just had a discussion with Jeff”
- Presenters attend sessions:
  - presents o in -> attend

- Listeners attend sessions:
  - listenTo o in -> attend

- Discussions happens in sessions:
  - speakWith o attend -> attend
Authors are interested in the topics
- \( \text{Inv(hasAuthor)} \circ \text{hasTopic} \rightarrow \text{interestedIn} \)

Listeners are interested in the topics
- \( \text{listenTo} \circ \circ \text{hasTopic} \rightarrow \text{interestedIn} \)
Query Examples

- **Recommending**
  - Talks of paper with interesting topic
    - \( t,r:- (t,p):of, (p,tp):hasTopic, (r,tp):interestedIn. \)
  - Researchers in the same session with same interest
    - \( r1,r2:- (r1,tp):interestedIn, (r2,tp):interestedIn, (r1,s):attend, (r2,s):attend. \)
Why Reasoning Matters

- **Inference**
  - Researcher activity information
    - The attend relation chain
  - Event information
    - Which ongoing talk is happening at what time in which venue given by whom?

- **Dynamics**
  - Researcher activities are happening in real time
  - Events are happening in real time
Setting Up (option 1)

- We use Eclipse IDE.
  - Put the SRTutorial folder into your Eclipse workspace
  - Create a Java project called SRTutorial in Eclipse. Eclipse will automatically use the source files inside the folder
  - Make sure the two jar files in the \lib folder are included in the build path of the project

- TBox: SR.owl;

- ABox: SRt0-SRt2.owl;

- Run the example
  - demo/Example/SRTutorialExample.java runs the example
  - demo/Example/SRTutorialJustifiedExample.java runs the example with explanations for each answer
Setting Up: For Those Who Have No IDE

- Run the example directly with the pre-compiled jar files
  - SRTutorialExample.jar runs the example
    - Execute “java –jar SRTutorialExample.jar” in console
  - SRTutorialJustifiedExample.jar runs the example with explanations for each answer
    - Execute “java –jar SRTutorialJustifiedExample.jar” in console
Example

- **t0:**
  - (Jeff, streamReasoning):presents,
  - (Ian, streamReasoning):listenTo,
  - (streamReasoning, SRTutorial):in,
  - (Jeff, ontology):interestedIn,
  - (Ian, ontology):interestedIn

- **Query:** who are active participants that interested in ontology?
  - Jeff
  - Ian

- They both attend the SRTutorial session and interested in ontology
Example

- **t1:**
  - (Jeff, ontology): interestedIn,
  - (Ian, ontology): interestedIn
  - (Ian, DLAduction): presents,
  - (Du, DLAduction): listenTo,
  - (DLAduction, Algorithm): in,
  - (Ian, Jeff): speaksWith,

- **Query:** who are active participants that interested in ontology?
  - Jeff
  - Ian

- They both attend the Algorithm session and interested in ontology
Example

- **t2:**
  - (Jeff, ontology):interestedIn
  - (Ian, ontology):interestedIn
  - (Ian, DLAbyduction):presents
  - (Du, DLAbyduction):listenTo
  - (DLAbyduction, Algorithm):in
  - (Du, ontology):interested

- Query: who are active participants that interested in ontology?
  - Ian
  - Du

- Du becomes interested in ontology
- Jeff needs to attend a meeting
Stream Reasoning For Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan
http://streamreasoning.org/sr4ld2013

OWL Reasoning and Stream Reasoning with LOD
Jeff Z. Pan
University of Aberdeen
http://homepages.abdn.ac.uk/jeff.z.pan/pages/