Stream Reasoning For Linked Data

Jeff Z. Pan and Emanuele Della Valle

Agenda

- Introduction to Linked Data and OWL 2 (90m)
- C-SPARQL: A Continuous Extension of SPARQL (90m)
- Stream Reasoning techniques for RDFS and RDFS++ (60m)
- Approximate Reasoning and Approximate Stream Reasoning for OWL2-DL (60m)
- Hands-on Session (60m)
### Applications: more than social networks
- Ontology editing
- Web service registry
- Ontology evolution/versioning
- Mobile semantic web
- Sensor network
- …

### Two streams
- to erase stream
- to add stream

---

**Example**

Suppose we want to find out if anyone is sharing some interest with David and following him on Twitter.

Potential changes:
- David’s interests
- Other’s interests
- David’s tweets
- Other’s retweets

```plaintext
interest \circ interest \subseteq shareInterest
tweet \circ tweet \subseteq follow
(David, SemanticWeb) : interest
(Alex, SemanticWeb) : interest
Alex \neq David
```

```
(David, Post₁) : tweet
(David, Post₂) : tweet
(Alex, Post₂) : retweet
(David, Post₃) : tweet
(Alex, Post₃) : retweet
(David, Post₄) : tweet
```
Ontology Stream

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

- Key task
  - Answer a set of test queries at each snapshot ontology \(O(i)\)

Example: Test Query

- Is there anyone sharing some interest with David and following him on twitter?
  - Retrieve instances of

\[
\neg \{\text{David}\} \sqcap \exists \text{shareInterest}, \{\text{David}\} \sqcap \exists \text{follow}, \{\text{David}\}
\]

- Test queries:
  - \((\text{David, Post}_1) : \text{tweet}\)
  - \((\text{Alex, Post}_2) : \text{retweet}\)
  - \((\text{David, Post}_3) : \text{tweet}\)
  - \((\text{Alex, Post}_3) : \text{retweet}\)
  - \((\text{David, Post}_4) : \text{tweet}\)

  - No
  - Yes, Alex
Two Approaches to Stream Reasoning

- Naïve reasoning
  - Re-computing everything on O(i+1)

- “Incremental” reasoning
  - Can we re-use results on O(i)?
    - Keep the preserved results
    - Eliminate expired results
    - Derive new results

Some Related Works

- Incremental Reasoning [Cuenca Grau et. al. 2010, Suntisrivaraporn 2008]:
  - High complexity, low change rate
  - Only deal with Ad(i), not Er(i)

- Belief contraction [Flouris et. al. 2006]
  - Very difficult in DL
  - Different from erasing in stream reasoning
    - Belief contraction can remove any entailed formulas
    - Stream reasoning ONLY removes ASSERTED formulas

- C-SPARQL: Continuous SPARQL [Barbieri et. al. 2010]:
  - Defining a query language for RDF data streams
Stream Reasoning: Existing Works

- 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

- Time-Windowed variant [Barbieri et. al. 2010b]
  - Suitable for fixed time-window size
  - Labelling the entailments with expiration time
  - Delete obsolete entailments w.r.t. their time labels

Limitations:
- Expressive power: RDF(S), rule-reducible DLs
- Relying on the fixed time window

Solution We Seek

- Expressive power
  - Supporting OWL2 DL ontology streams

- Less/No prior knowledge
  - Flexible addition and removal of axioms

- Efficiency
  - Low computational complexity
  - Pragmatic usefulness and implementation

- Scalability
  - Absolute volume of the stream
  - Relative volume of update
Approach Overview

- How to improve efficiency and scalability?
  - Reducing complexity with approximation

- How to re-use results on O(n)?
  - Follow the DRed strategy
  - Maintain the derivation relations among entailments, to
    - Identify the preserved results
    - Identify the expired results

- How to perform incremental reasoning
  - based on justifications

Two Approaches to Scalable Reasoning

- Materialisation
  - Pro: usually more efficient
  - Con: doesn't work if data change all the time

- Query rewriting
  - Pro: no need for materialisation
  - Con: only work for languages with limited expressive power
Approximate Reasoning

Approach 1: Knowledge Compilation [Selman and Kautz, 1996]
- Lower-bound $KB_{lb}$: $M(KB_{lb}) \subseteq M(KB)$ (hence $KB_{lb} \models KB$)
  - Greatest lower-bound: no $KB'$ such that $M(KB_{lb}) \subseteq M(KB') \subseteq M(KB)$
- Upper-bound $KB_{ub}$: $M(KB) \subseteq M(KB_{ub})$ (hence $KB \models KB_{ub}$)
  - Least upper-bound: no $KB'$ such that $M(KB) \subseteq M(KB') \subseteq M(KB_{ub})$

Approach 2: Language Weakening
- Inputs (query, ontology)

Approach 3: Approximate Deduction
- Algorithm

Outputs (Quality Control)
- soundness preserving
- completeness preserving
Semantic Approximation [PaTh-AAAIAI2007]

- Least upper bound approximation for QL ES(O, OWL 2 QL) of an OWL2 DL O is finite and unique.

- **Theorem 1**: Given an ontology O, a conjunctive query q(X) and an evaluation [X→S], if ES(O, OWL 2 QL) |= q[X→S], then O |= q[X→S].

- **Theorem 2**: Given an ontology O_S, a database-style conjunctive query q(X) without non-distinguished variables and an evaluation [X→S], ES(O_S, OWL 2 QL) |= q[X→S] iff O_S |= q[X→S].

Evaluation: Quality vs. Performance

- **Lehigh University Benchmark**

  **OWL 2 DL/ SHIQ reasoners [Motik and Sattler, 2006]**
  - Pellet and Racer: unable to complete queries 1, 2 and 3 when more than 2 universities
  - KAON2 can answer queries for up to 4 universities

- **Syntactic approximation systems [Guo et al., 2004]**
  - OWLJessKB can only handle 1 university
    - All results are covered
    - Returned **incorrect** answers for queries 4, 6, 8 and 12
  - DLDB can handle 50 universities on all queries but query 2
    - Returned **incomplete** answers for queries 11 and 13
Evaluation: Query Answering (III)
(Quill – QL reasoner in TrOWL)

- Quill/ONTOSEARCH2 can handle at least 50 universities
  - Sound and complete results for all queries
- DLDB vs. Quill/ONTOSEARCH2

![Graph showing performance comparison between DLDB and Quill/ONTOSEARCH2](image)

Faithful Syntactic Approximate Reasoning
[AAAI2010]

- 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</td>
<td>\bot</td>
</tr>
<tr>
<td>( C \subseteq D )</td>
<td>( r \subseteq s, r_1 \circ \ldots \circ r_n \subseteq s )</td>
</tr>
<tr>
<td>( a : C )</td>
<td>( (a, b) : r )</td>
</tr>
</tbody>
</table>

N2EXPTIME-complete | PTIME-complete
TBox approximation
- Directly represent non-OWL2-EL concepts with fresh named concepts
  - E.g., $\forall r.C \text{ subClassOf } D \Rightarrow A_{\forall r.C} \text{ subClassOf } D$
- Maintain semantic relations for these named concepts
  - complementary relations
  - cardinality relations

ABox approximation
- Internalise ABox into TBox
  - E.g., $C(a) \Rightarrow \{a\} \text{ subClassOf } C$
  - $r(a,b) \Rightarrow \{a\} \text{ subClassOf } \exists r.\{b\}$, ...

Reasoning
- Using additional tractable completion rules to recover the semantics

Extra completion Rules, e.g.
- Handling complement
  - E.g. $A \text{ subClassOf } B \Rightarrow \neg B \text{ subClassOf } \neg A$
- Handling cardinality
  - E.g. $A \text{ subClassOf } \geq 3 \Rightarrow A \text{ subClassOf } \geq 2 \Rightarrow B$
- Soundness preserving and tractable
Demo 1: TBox Reasoning

- Demo plan: Oxford Benchmark and some related ontologies
  1. classification of DLP397 in Pellet: time consuming
  2. classification of DLP397 in REL: efficient
  3. classification of Trento1 in REL: efficient
  4. classification of Trento1 in Pellet: time consuming

Evaluations for the Oxford Benchmarks (REL – EL reasoner in TrOWL)
Truth Maintenance System

- A directed graph
  - Nodes: axioms / entailments
  - Edges: derivation relations among axioms / entailments
  - All entailments are reachable from their justifications
    - Easy to identify impacted entailments

Stream Reasoning 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
Features and Optimisations

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

- **Disadvantages & optimisations**
  - 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
    - Syntactic approximation

Syntactic Approximated-TMS

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

Original Ontology

```
interest ∩ interest' ⊆ shareInterest
```

Approximated Ontology

```
{Alex} ∩ interest ∩ SemanticWeb
```

```
{Alex} ∩ ∅
```

```
shareInterest ⊆ {Alex} ∩ ∅
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ ∅ ⊆ interest ∩ ∅
```

```
interest ∩ ∅ ⊆ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest ⊆ {Alex} ∩ ∅
```

```
{Alex} ∩ ∅ ⊆ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```

```
{Alex} ∩ ∅ ⊆ shareInterest
```
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

Demo 2: Stream reasoning with LUBM

- Demo plan:
  - 1. Naive approach
  - 2. Incremental reasoning approach
Evaluation

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

Stream Reasoning in Conferences

- 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"
Ontology

- Presenters attend sessions:
  - presents o in -> attend

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

- Discussions happens in sessions:
  - speakWith o attend -> attend

Ontology (cont.)

- Authors are interested in the topics
  - Inv(hasAuthor) o hasTopic -> interestedIn

- Listeners are interested in the topics
  - listenTo o of o hasTopic -> interestedIn
Query Examples

- **Recommendation queries**
  - Talks that researchers are interested in
    \[ q_1(t,r) :- (t,p):of, (p,tp):hasTopic, (r,tp):interestedIn. \]
  - Researchers in the same session with common interests
    \[ q_2(r_1,r_2) :- (r_1,tp):interestedIn, (r_2,tp):interestedIn, (r_1,s):attend, (r_2,s):attend. \]

NBox Reasoning:
An Application of Stream Reasoning

- **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>Name</th>
<th>Vegetarian</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Yuting</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Jek</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yuan</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
NBox Reasoning

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

Demo: NBox Reasoning

- Local closed world reasoning in DMIL
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

SQL connection 1

Connection

SQL data port type

Query 1 SQL data

Output Port

Which input port can NOT be used here?
Local closed world reasoning in DMIL

- Data source port type
- SQL port type
- Restricted connection
- Occupied
- Merge data 1
- SQL connection 1
- Class connection
- Query 1 SQL data
- Which input port can NOT be used here?

- Valid input port
- Input Port
- Output Port
- Which input port can be used here?

Demo: NBox Reasoning
Demo plan
- Load ontology
- Put “input ports that can not be used” (Test) into NBox
- Perform LCW reasoning with NBox
  - Retrieve “input ports that can be used” (Input AND (NOT Test))

TrOWL is a tractable OWL2 reasoning infrastructure developed at Aberdeen [ESWC2010]
- It supports many pluggable reasoners
- It works with OWL API, Protégé 4, Jena and Semantic Mediawiki
- Configuration through property files and feature modelling:
  - Performance required
  - Determine the best reasoner for a specific task

Quality guaranteed transformations
- Faithful approximate reasoning
  - Quill: OWL 2 DL -> OWL 2 QL (semantic approximation) [AAA12007]
  - REL: OWL 2 DL -> OWL 2 EL (syntactic approximation) [AAA12010]
- Divide and conquer
  - Modularisation [ISWC2009]
  - Forgetting [ESWC2008, ISWC2009b]

Coming up soon: SPARQL1.1, Stream Reasoning
Approximate reasoning and TMS play key roles in OWL 2 DL stream reasoning

**Hand-on Session**

- 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, DLabduction): presents,
  - (Du, DLabduction): listenTo,
  - (DLabduction, 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, DLabduction): presents,
  - (Du, DLabduction): listenTo,
  - (DLabduction, Algorithm): in,
  - (Du, ontology): interested

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

- Du becomes interested in ontology
- Jeff could left the session
Acknowledgement

- Aidan Hogan
- Ian Horrocks
- Nophadol Jekjantuk
- Axel Polleres
- Yuan Ren
- Edward Thomas
- Yuting Zhao

Stream Reasoning For Linked Data

Jeff Z. Pan and Emanuele Della Valle