Efficient Identification of Implicit Facts in Incomplete OWL2-EL Knowledge Bases

John Liagouris, Manolis Terrovitis

IMIS - Research Center “Athena”
Web Ontology Language (OWL)

- Extends the RDF Schema (RDFS)
  - `rdfs:Class`, `rdfs:Property`

- Complex class expressions
  - `Woman ≡ Person \cap Female`

- Complex property expressions
  - `hasMother \circ hasSister \sqsubseteq hasAunt`

- Property characteristics
  - We can define a RDF property like “isSiblingWith” as symmetric and transitive
Web Ontology Language (OWL)

- The semantics of OWL imply additional knowledge, i.e., new RDF triples

\[ \langle \text{mary rdf:type Woman}\rangle + \text{“} \text{Woman} \equiv \text{Person} \cap \text{Female} \text{”} \]

\[ \downarrow \]

\[ \langle \text{mary rdf:type Person}\rangle \]
\[ \langle \text{mary rdf:type Female}\rangle \]

- Such “hidden” triples cannot be queried directly with traditional SPARQL engines
- Reasoning is needed
Complexity of Reasoning in OWL

● Exponential for OWL in general

● Three tractable fragments:
  ○ OWL2-QL
  ○ OWL2-EL
  ○ OWL2-RL

● Each fragment poses different restrictions in the syntax of OWL
Problem

- We are given a large collection of OWL2-EL axioms and a set of inference rules

- **Goal**: Infer all axioms that are implied by the rules

- **How**: Apply all rules to the collection of axioms exhaustively till no new axioms are produced (fix-point)
Running Example

We are given a collection of OWL2-EL axioms of the form $X \sqsubseteq Y$:

1. $\text{InfectedWithVirusA} \sqcap \text{NotVaccinated} \sqsubseteq \text{Ill}$
2. $\exists \text{Vaccinated}.\text{VaccineTypeX} \sqsubseteq \text{NotVaccinated}$
3. $\text{Vaccinated1994} \sqsubseteq \exists \text{Vaccinated}.\{\text{va}\}$
4. $\{\text{va}\} \sqsubseteq \text{VaccineTypeX}$
5. $\{\text{john}\} \sqsubseteq \text{Vaccinated1994}$
6. $\{\text{john}\} \sqsubseteq \text{InfectedWithVirusA}$
Running Example

2. \( \exists \text{Vaccinated}. \text{VaccineTypeX} \sqsubseteq \text{NotVaccinated} \)
3. \( \text{Vaccinated1994} \sqsubseteq \exists \text{Vaccinated}. \{\text{va}\} \)
4. \( \{\text{va}\} \sqsubseteq \text{VaccineTypeX} \)

From axioms 3, 4 and 2 \( \rightarrow \) 7. \( \text{Vaccinated1994} \sqsubseteq \text{NotVaccinated} \)
Running Example

5. \{john\} \sqsubseteq\text{Vaccinated1994}

From axioms 3, 4 and 2 \rightarrow 7. \text{Vaccinated1994} \sqsubseteq\text{NotVaccinated}

From axioms 5 and 7 \rightarrow 8. \{john\} \sqsubseteq\text{NotVaccinated}
Running Example

1. InfectedWithVirusA ⊓ NotVaccinated ⊑ Ill

6. {john} ⊑ InfectedWithVirusA

From axioms 3, 4 and 2 → 7. Vaccinated1994 ⊑ NotVaccinated

From axioms 7 and 5 → 8. {john} ⊑ NotVaccinated

From axioms 6, 8 and 1 → 9. {john} ⊑ Ill
Challenges

- Inference rules for OWL2-EL are complex and mutually recursive (each one affects the other).
- The collection of axioms does not always fit in main-memory.
- The inference requires repetitive scans of the axioms:
  - The problem becomes I/O bounded.
Our Contribution

● All existing rule engines apply the inference rules sequentially

● They scan the ontology on a per-rule basis

● We define a uniform access pattern which allows for the in-bulk application of many rules within the same scan
Graph Model

2. \( \exists \text{Vaccinated}.\text{VaccineTypeX} \sqsubseteq \text{NotVaccinated} \)
Graph Model

4. \( \{va\} \sqsubseteq \text{VaccineTypeX} \)
Graph Model

3. $\text{Vaccinated1994} \subseteq \exists \text{Vaccinated.}\{\text{va}\}$
Graph Model

5. \( \{\text{john}\} \sqsubseteq \text{Vaccinated1994} \)
Graph Model

6. \{john\} ⊈ InfectedWithVirusA
Graph Model

1. \( \text{InfectedWithVirus} \cap \text{NotVaccinated} \subseteq \text{Ill} \)
Inference on the Graph

1. \( \text{InfectedWithVirusA} \cup \text{NotVaccinated} \subseteq \text{Ill} \)
1. $\text{InfectedWithVirusA} \sqcap \text{NotVaccinated} \sqsubseteq \text{Ill}$
Inference on the Graph

1. \( \text{InfectedWithVirusA} \sqcap \text{NotVaccinated} \subseteq \text{Ill} \)
Idea of the Algorithm

- Store the graph in a way that allows efficient lookups in the neighbourhood of each node
- Keep track of the changes made in the graph
- At each subsequent step check only the neighbourhoods affected from the previous step
Experiments

- **Real ontologies**
  - SNOMED CT
  - GALEN8

- **Synthetic ontologies**
  - Ontologies of different sizes whose graphs are isomorphic to the graphs of SNOMED CT and GALEN
  - Ontologies of different sizes by increasing the number of labeled edges per node
Experiments on Real Data (SNOMED CT)

**BRA**: Batch Rule Application  
**ORT**: One Rule at a Time  
**BRA-M**: Main-memory version of BRA  
**YAP, XSB**: Prolog-based systems  
**DLV, LogicBlox**: Datalog engines
Experiments on Real Data (GALEN8)

**BRA**: Batch Rule Application

**ORT**: One Rule at a Time

**BRA-M**: Main-memory version of BRA

**YAP, XSB**: Prolog-based systems

**DLV, LogicBlox**: Datalog engines
Experiments on Real Data
(Optimizations)

**BRA**: Batch Rule Application
**BRA-A**: BRA on the schema of ORT
**SN**: BRA without optimizations
**ORT**: One Rule at a Time
Experiments on Synthetic Data (Isomorphic Graphs of SNOMED CT)
Experiments on Synthetic Data (Isomorphic Graphs of GALEN8)
Experiments on Synthetic Data (Increasing Node Degree for GALEN8)
OWL2-EL Syntax and Semantics

● Intersection of Classes
  o Father ≡ Male ∩ Parent
  o “Father is the class of all individuals which are of type Male and also of type Parent”

● Existential Restrictions
  o Grandparent ≡ ∃ hasChild. Parent
  o “Grandparent is the class of all individuals which are linked through property “hasChild” with an individual of type Parent”
OWL2-EL Syntax and Semantics

- Reflexivity
  - Narcissus $\equiv \exists \text{likes}.\text{Self}$
  - “Narcissus is the class of all individuals which are linked through property ‘likes’ with themselves”

- Property axioms
  - $\text{hasSister} \sqsubseteq \text{siblingWith}$
  - $\text{hasMother} \circ \text{hasSister} \sqsubseteq \text{hasAunt}$
OWL2-EL Syntax and Semantics

- Singleton Nominals (individuals/instances)
  - \{mary\} ⊑ Woman ↔ <mary rdfs:type Woman>
  - \{mary\} ⊑ ∃siblingWith.{tom} ↔ <mary siblingWith tom>
  - \{mary\} ≡ \{maria\} ↔ <mary owl:sameAs maria>

- These are the actual data modeled with the ontology
Why OWL2-EL??

- It is widely used in Life Sciences
- Large OWL2-EL ontologies like SNOMED CT have become vital parts of the Health Information Systems in many countries
- It supports the definition of Tuple-Generating Dependencies (TGDs)
  - Suitable for data integration scenarios