ISWC 2017 Tutorial:
Semantic Data Management in Practice

Part 6: Automated reasoning

Olaf Hartig
Linköping University
olaf.hartig@liu.se
@olafhartig

Olivier Curé
University of Paris-Est Marne la Vallée
olivier.cure@u-pem.fr
@oliviercure

Goals

- Apprehend reasoning in a Knowledge base management context
- Understand the pros and cons of materialization and query reformulation
- Select to proper production-ready RDF store based on your reasoning needs

Overview

- Reasoning introduction
- Semantic Web ontologies expressiveness
- Reasoning methods: materialization & query reformulation
- Our production ready systems and reasoning
- Demo

Reasoning

- The act of deriving implicit data from explicit one
- In a semantic context, deriving implies the use of logic-based processing
- This is based on a set of ontologies which are representing knowledge on the domain of discourse

Reasoning in Data and Knowledge Bases

- Database management systems natively do not support reasoning
  - Some external solutions are possible but rare in production, e.g., XSB a Logic Programming systems that can connect to most RDBM systems

- Knowledge bases natively support inferences
  - All our 7 production-ready systems support reasoning

Overview

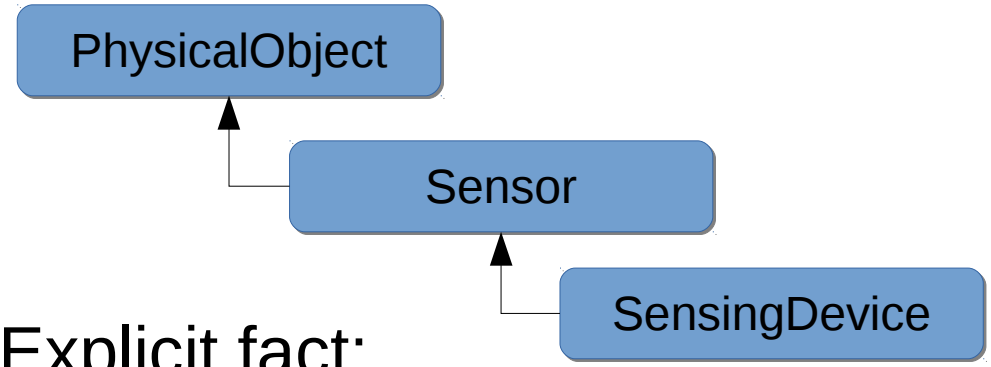
- Reasoning introduction
- Semantic Web ontologies expressiveness
- Reasoning methods: materialization & query reformulation
- Our production ready systems and reasoning
- Demo

Supported Ontology Languages

- KB are concerned with a trade-off between ontology expressiveness and computational difficulty
- Intuitively, ontology languages with a tractable complexity are more amenable to reasoning in a KB system than ones with an exponential complexity
- The ‘tractables’ are:
  - RDFS
  - RDFS++ (RDFS with some OWL constructs)
  - OWLQL
  - OWLRL
  - OWL Horst
- Non tractable but decidable:
  - OWLDL

RDFS Example

- Concept hierarchy (extract from SSN):



- Explicit fact:

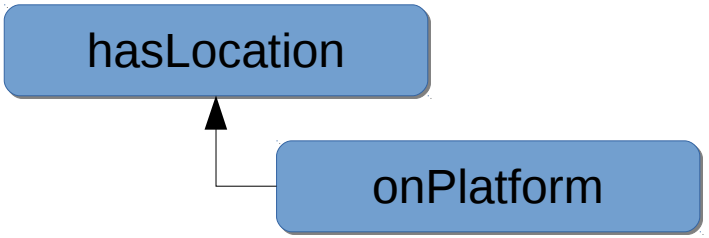
```
QBE04 rdf:type SensingDevice
```

- Implicit facts

```
QBE04 rdf:type Sensor
QBE04 rdf:type PhysicalObject
```

RDFS Example

- Property hierarchy (extract from SSN):



- Explicit fact:

```
QBE04 ssn:onPlatform Platform#2
```

- Implicit fact

```
QBE04 ssn:hasLocation Platform#2
```

RDFS++ Example

- RDFS + sameAs + transitive and inverseOf propertiesKB
- In SSN

```
hasLocation inverseOf attachedSystem
```

- Explicit fact

```
QBE04 ssn:onLocation Platform#2
```

- Implicit fact

```
PlatForm#2 ssn:attachedSystem QBE04
```

RDFS++ Example

- RDFS + sameAs + transitive and inverseOf properties
- Consider

```
QBE04 sameAs QBS06
```

- Explicit facts

```
QBE04 ssn:onLocation Platform#2
QBE04 rdf:type SensingDevice
```

- Implicit facts

```
QBS06 ssn:onLocation Platform#2
QBS06 rdf:type SensingDevice
```

RDFS++ Example

- RDFS + sameAs + transitive and inverseOf properties
- Consider

```
QBE04 sameAs QBS06
```

- Explicit facts

```
QBE04 ssn:onLocation Platform#2
QBE04 rdf:type SensingDevice
```

- Implicit facts

```
QBS06 ssn:onLocation Platform#2
QBS06 rdf:type SensingDevice
QBS06 rdf:type Sensor
QBS06 rdf:type PhysicalObject
PlatForm#2 ssn:attachedSystem QBE04
```

RDFS++ Example

- RDFS + sameAs + transitive and inverseOf properties
- Consider

```
hasPart rdf:type TransitiveProperty
```

- Explicit facts

```
Entity1 hasPart Entity2
Entity2 hasPart Entity3
```

- Implicit fact

```
Entity1 hasPart Entity3
```

Overview

- Reasoning introduction
- Semantic Web ontologies expressiveness
- Reasoning methods: materialization & query reformulation
- Our production ready systems and reasoning
- Demo

Reasoning Methods

- Two main reasoning methods are possible
  - Materialization
  - Query rewriting

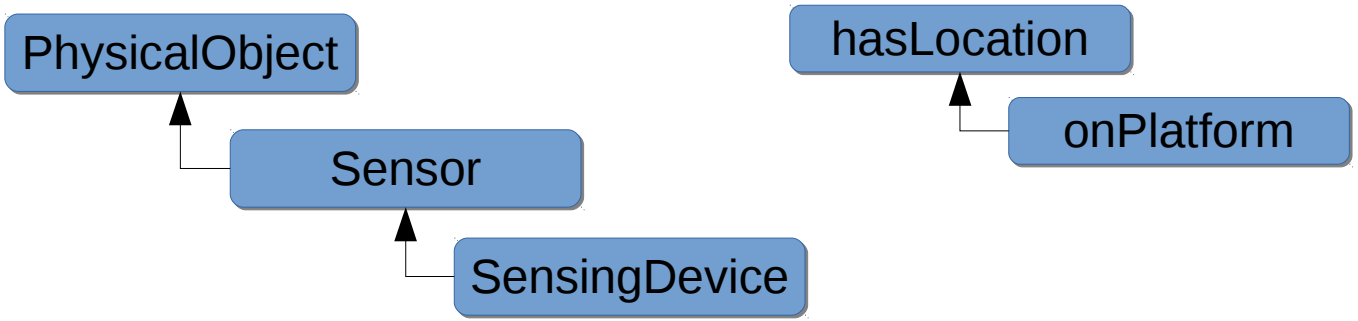
Materialization

- Make explicit all inferences in the store

- Pros:
  - Efficient query processing (no reasoning at query runtime)
- Cons:
  - Slow data loading
  - Data volume expansion
  - Tricky update management

- aka forward reasoning or closure

Materialization Example



- Explicit facts:

```
QBE04 ssn:onPlatform Platform#2
QBE04 rdf:type SensingDevice
```

- The following facts would be added:

```
QBE04 rdf:type Sensor
QBE04 rdf:type PhysicalObject
QBE04 ssn:hasLocation Platform#2
```

Query Rewriting

- Reformulate the original query such that all answers can be retrieved

- Pros:
  - No preprocessing overhead
  - No expansion of stored data volume
  - Easy update management
- Cons:
  - Slow query processing due to cost of reasoning at query runtime

- aka Backward reasoning or query reformulation

Query Rewriting Example

- Original query:

```
SELECT ?x WHERE { ?x rdf:type
PhysicalObject}
```

- Reformulated query:

```
SELECT ?x WHERE { ?x rdf:type
PhysicalObject}
UNION
SELECT ?x WHERE { ?x rdf:type Sensor}
UNION
SELECT ?x WHERE { ?x rdf:type
SensingDevice}
```

Overview

- Reasoning introduction
- Semantic Web ontologies expressiveness
- Reasoning methods: materialization & query reformulation
- Our production ready systems and reasoning
- Demo

Prod-ready Systems and reasoning

Triple store	Materialization	Query rewriting
Allegrograph	OWLRL	RDFS++, Prolog
Blazegraph	RDFS, OWL Lite	
GraphDB	RDFS, OWL Horst, OWLRL, OWLQL	
MarkLogic		RDFS, RDFS++, OWL Horst
Oracle	RDFS, OWLRL, OWLQL	
Stardog	All OWL2	
Virtuoso		RDFS++

Overview

- Reasoning introduction
- Semantic Web ontologies expressiveness
- Reasoning methods: materialization & query reformulation
- Our production ready systems and reasoning
- Demo

Demo

- Using Blazegraph
- N3 files containing the SSN ontology and an extract of facts. Total of approx. 1200 triples
- Some Java code using the RDF4J API, any Java IDE (here IntelliJ)
- Different Blazegraph property files to highlight reasoning aspects