Ontology Semantics, Repair and Enrichment
(with a Focus on Linked Data Knowledge Bases)

Jens Lehmann, Lorenz Bühmann

Creating Knowledge out of Interlinked Data

AKSW

2011-09-15
Personal Introduction

Jens Lehmann

- PhD Uni Leipzig 2006-2010
- Head of Machine Learning and Ontology Engineering Group (MOLE) at AKSW since 2010
- Software/Research-Projects: DL-Learner, DBpedia, ORE, LinkedGeoData, AutoSPARQL, ReDD, SAIM, NLP2RDF

Lorenz Bühmann

- Studied Computer Science at Uni Leipzig 2006 - 2011
- PhD Uni Leipzig since 2011
- Software/Research-Projects: ORE, DL-Learner, AutoSPARQL
Outline

1. **OWL 2 Structure Overview**
2. **OWL 2 Semantics**
3. **Ontology Debugging and Repair via ORE**
4. **Ontology Enrichment via DL-Learner/ORE**
5. **Examples and Demo**
6. **Related Tools**
7. **Conclusions and Future Work**
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
OWL 1 W3C Recommendation since 2004 and OWL 2 since 2009

Semantic fragment of FOL
OWL (2) General

- OWL 1 W3C Recommendation since 2004 and OWL 2 since 2009
- Semantic fragment of FOL
- Variants of OWL 1: OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full
- Variants of OWL 2: OWL 2 DL $\subseteq$ OWL 2 Full
- Profiles of OWL 2: OWL 2 QL, OWL 2 EL++, OWL 2-RL DL, OWL 2-RL Full

OWL DL is decidable and corresponds to description logic SHOIN ($D$)
OWL 2 DL corresponds to SROIQ ($D$)

W3C documents contain more details than discussed here
• **OWL 1 W3C Recommendation** since 2004 and OWL 2 since 2009
• Semantic fragment of FOL
• Variants of OWL 1: OWL Lite $\subseteq$ OWL DL $\subseteq$ OWL Full
• Variants of OWL 2: OWL 2 DL $\subseteq$ OWL 2 Full
• Profiles of OWL 2: OWL 2 QL, OWL 2 EL++, OWL 2-RL DL, OWL 2-RL Full
• OWL DL is decidable and corresponds to description logic $\mathcal{SHOIN}(D)$
• OWL 2 DL corresponds to $\mathcal{SROIQ}(D)$
• W3C documents contain more details than discussed here
we use Manchester Syntax and Description Logic (DL) syntax in this presentation

DL syntax:
Student ⊑ Person

Manchester syntax (will be introduced when needed):
Class: Student SubClassOf: Person
As in OWL 1:
- Ontology = Set of axioms (+ Head)
- 1 Axiom = 1...n RDF Triple

Physical location has to correspond to version URI (if it exists) and current version has to be found at ontology URI (if it exists)
The diagram illustrates the relationships between different entities and data types in the context of the OWL 2 - Entities framework.

- **Class**: Represents a collection of entities sharing common characteristics.
- **ObjectProperty**: Specifies a relationship between entities.
- **DataProperty**: Specifies a property that can be assigned to an entity with a data type.
- **AnnotationProperty**: Specifies a property that can be assigned to an entity as a metadata.
- **Datatype**: Represents a type of data that can be assigned to an entity.
- **NamedIndividual**: Represents a specific, named entity.
- **AnonymousIndividual**: Represents an entity without a specific name.
- **Literal**: Represents a value that is not further specified.
- **xsd:string** and **xsd:integer**: Represent different data types.

The diagram includes symbols for logical operations such as **⊤** (true) and **⊥** (false). It also highlights the concept of an **arity of a datatype must be one** restriction and a property named **rdfs:label**.

- **isCapitalOf**: Represents a property indicating that Leipzig is the capital of John.
- **hasAge**: Indicates an unspecified property of the entity John.
- **City**: Represents a class or entity.

The diagram is part of a larger discussion on Repair and Enrichment, as indicated by the reference to Lehmann, Bühmann (Univ. Leipzig) and the date 2011-09-15.
OWL 2 – Class Expressions

LivingPerson ⊓¬ (Teenager ⊔ Adult) {Germany, France}
∀hasPet.Dog

∃birthPlace.\{Leipzig\}

foaf:Person \bigcap \existsfoaf:image
OWL 2 – Class Expressions

ObjectMaxCardinality

ObjectMinCardinality

ObjectExactCardinality

Exam \sqsubseteq 2 examiner

Exam \sqsupseteq 3 topic

ClassExpression

ObjectPropertyExpression

0..1

0..1

0..1

classExpression

classExpression

classExpression

objectPropertyExpression

objectPropertyExpression

objectPropertyExpression
foaf:homepage ⊆ foaf:isPrimaryTopicOf

foaf:img
domain: foaf:Person
range: foaf:Image

foaf:depiction ≡ foaf:depicts\(^{-1}\)
APIs: OWL API, KAON2
Editors: Protégé, TopBraid
Reasoning:
  - OWL 2 DL: Pellet, FaCT++
  - OWL 2 EL: CEL
  - OWL 2 QL: QuONto, Owlgres
  - OWL 2 RL: Oracle 11g
...
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
Description Logics (DLs)

- family of formal knowledge representation languages
- fragment of FOL
- mostly decidable
- relatively expressive
- created from attempts to formalize semantic networks
- intuitive syntax
- variable-free

W3C-Standard OWL DL is based on Description Logic $SHOIN(D)$ (OWL 2 DL based on $SROIQ(D)$)

we discuss $ALC$ for explaining OWL/DL semantics
Basic components:

- **class** names (also referred to as concepts)
- **role** names
- **individual** names (also referred to as objects)

knowledge base $\equiv$ set of axioms

Axioms for instance data:

- $\text{Man}(\text{Bob})$
  $\equiv$ individual $\text{Bob}$ belongs to class $\text{Man}$

- $\text{hasPet}(\text{Bob}, \text{Tweety})$
  $\equiv$ $\text{Bob}$ has a pet $\text{Tweety}$
Child $\sqsubseteq$ Person

- "Every child is a person."
- corresponds to $(\forall x)(\text{Child}(x) \rightarrow \text{Person}(x))$
- corresponds to rdfs:subClassOf

Man $\equiv$ MaleHuman

- "Man are exactly the male humans."
- corresponds to $(\forall x)(\text{Man}(x) \leftrightarrow \text{MaleHuman}(x))$
- corresponds to owl:equivalentClass
Conjunction $\sqcap$ corresponds to $\text{owl:intersectionOf}$

Disjunction $\sqcup$ corresponds to $\text{owl:unionOf}$

Negation $\neg$ corresponds to $\text{owl:complementOf}$

Example:

$\text{Professor} \sqsubseteq (\text{Person} \sqcap \text{UniversityMember}) \sqcup (\text{Person} \sqcap \neg \text{PhDStudent} \sqcap \text{Postgraduate})$

Predicate logic: $(\forall x) (\text{Professor}(x) \rightarrow ((\text{Person}(x) \land \text{UniversityMember}(x))) \lor \text{Person}(x) \land \neg \text{PhDStudent}(x) \land \text{Postgraduate}(x))$
Exam $\subseteq \forall \text{hasExaminor}.\text{Professor}$

- "Every exam has only professors as examinor."
- $(\forall x)(\text{Exam}(x) \rightarrow (\forall y)\text{hasExaminor}(x, y) \land \text{Professor}(y)))$
- corresponds to `owl:allValuesFrom`

Exam $\subseteq \exists \text{hasExaminor}.\text{Person}$

- "Every exam has at least one examinor."
- $(\forall x)(\text{Exam}(x) \rightarrow (\exists y)(\text{hasExaminor}(x, y) \land \text{Person}(y)))$
- corresponds to `owl:someValuesFrom`
The following syntax rules create classes in $\mathcal{ALC}$. Here $A$ is an atomic class and $R$ a role

$C, D \rightarrow A | \top | \bot | \neg C | C \cap D | C \cup D | \forall R.C | \exists R.C$

- An $\mathcal{ALC}$-TBox consists of expressions of type $C \sqsubseteq D$ and $C \equiv D$, where $C, D$ are classes.
- An $\mathcal{ALC}$-ABox consists of expressions of type $C(a)$ and $R(a, b)$, where $C$ is a complex class, $R$ is a role and $a, b$ are individuals.
- An $\mathcal{ALC}$- knowledge base consists of a ABox and a TBox.
we define the model-theoretic semantics of $\mathcal{ALC}$ (i.e. entailment is defined by interpretations)

an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of

- a set $\Delta^\mathcal{I}$, called domain and
- a function $\cdot^\mathcal{I}$, which maps from
  - individual names $a$ to elements of the domain $a \in \Delta^\mathcal{I}$
  - class names $C$ to set of domain elements $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
  - role names $R$ to set of pairs of domain elements $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
will be extended to complex classes:

\[ \top^\mathcal{I} = \triangle^\mathcal{I} \]
\[ \bot^\mathcal{I} = \emptyset \]
\[ (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I} \]
\[ (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I} \]
\[ (\forall R \cdot C)^\mathcal{I} = \{ x \mid \forall (x, y) \in R^\mathcal{I} \rightarrow y \in C^\mathcal{I} \} \]
\[ (\exists R \cdot C)^\mathcal{I} = \{ x \mid \exists (x, y) \in R^\mathcal{I} \text{ mit } y \in C^\mathcal{I} \} \]
\[ (\neg C)^\mathcal{I} = \triangle^\mathcal{I} \setminus C^\mathcal{I} \]
... and finally to axioms:

- $C(a)$ is satisfied in $\mathcal{I}$, if: $a^\mathcal{I} \in C^\mathcal{I}$
- $R(a, b)$ is satisfied in $\mathcal{I}$, if: $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$
- $C \sqsubseteq D$ is satisfied in $\mathcal{I}$, if: $C^\mathcal{I} \subseteq D^\mathcal{I}$
- $C \equiv D$ is satisfied in $\mathcal{I}$, if: $C^\mathcal{I} = D^\mathcal{I}$

Interpretations, which satisfy an axiom (resp. a set of axioms), are called models.
Interpretations - Examples

\[ \text{TBox } \mathcal{T} : \quad \text{Man} \equiv \neg \text{Woman} \sqcap \text{Person} \]
\[ \quad \text{Woman} \sqsubseteq \text{Person} \]
\[ \quad \text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top \]

\[ \text{ABox } \mathcal{A} : \quad \text{Man}(\text{STEPHEN}). \]
\[ \quad \neg \text{Man}(\text{MONICA}). \]
\[ \quad \text{Woman}(\text{JESSICA}). \]
\[ \quad \text{hasChild}(\text{STEPHEN}, \text{JESSICA}). \]

For all following interpretations \( \mathcal{I} \):

- domain \( \Delta^\mathcal{I} = \{ \text{MONICA, JESSICA, STEPHEN} \} \)
- objects are mapped to themselves (\( a^\mathcal{I} = a \))
Interpretations - Examples

TBox $\mathcal{T}$:  
$\text{Man} \equiv \neg \text{Woman} \sqcap \text{Person}$
$\text{Woman} \sqsubseteq \text{Person}$
$\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top$

ABox $\mathcal{A}$:  
$\text{Man(STEPHEN)}.$
$\neg \text{Man(MONICA)}.$
$\text{Woman(JESSICA)}.$
$\text{hasChild(STEPHEN, JESSICA)}.$

$\text{Man}^{\mathcal{I}_1} = \{ \text{JESSICA, STEPHEN} \}$
$\text{Woman}^{\mathcal{I}_1} = \{ \text{MONICA, JESSICA} \}$
$\text{Mother}^{\mathcal{I}_1} = \emptyset$
$\text{Person}^{\mathcal{I}_1} = \{ \text{JESSICA, MONICA, STEPHEN} \}$
$\text{hasChild}^{\mathcal{I}_1} = \{ (\text{STEPHEN, JESSICA}) \}$
Interpretations - Examples

\[ \text{TBox } \mathcal{T} : \quad \text{Man} \equiv \neg \text{Woman} \sqcap \text{Person} \]
\[ \mathcal{I}_1 \not\models \mathcal{T} \quad \text{Woman} \sqsubseteq \text{Person} \]
\[ \text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top \]

\[ \text{ABox } \mathcal{A} : \quad \text{Man}(\text{STEPHEN}). \]
\[ \mathcal{I}_1 \models \mathcal{A} \quad \neg \text{Man}(\text{MONICA}). \]
\[ \text{Woman}(\text{JESSICA}). \]
\[ \text{hasChild}(\text{STEPHEN}, \text{JESSICA}). \]

\[ \text{Man}^{\mathcal{I}_1} = \{ \text{JESSICA}, \text{STEPHEN} \} \]
\[ \text{Woman}^{\mathcal{I}_1} = \{ \text{MONICA}, \text{JESSICA} \} \]
\[ \text{Mother}^{\mathcal{I}_1} = \emptyset \]
\[ \text{Person}^{\mathcal{I}_1} = \{ \text{JESSICA}, \text{MONICA}, \text{STEPHEN} \} \]
\[ \text{hasChild}^{\mathcal{I}_1} = \{ \text{(STEPHEN, JESSICA)} \} \]
Interpretations - Examples

TBox $\mathcal{T}$:
- $\text{Man} \equiv \neg \text{Woman} \sqcap \text{Person}$
- $\text{Woman} \sqsubseteq \text{Person}$
- $\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top$

ABox $\mathcal{A}$:
- $\text{Man}(\text{STEPHEN})$.
- $\neg \text{Man}(\text{MONICA})$.
- $\text{Woman}(\text{JESSICA})$.
- $\text{hasChild}(\text{STEPHEN}, \text{JESSICA})$.

$$
\begin{align*}
\text{Man}^{I_2} &= \{\text{STEPHEN}\} \\
\text{Woman}^{I_2} &= \{\text{JESSICA, MONICA}\} \\
\text{Mother}^{I_2} &= \emptyset \\
\text{Person}^{I_2} &= \{\text{JESSICA, MONICA, STEPHEN}\} \\
\text{hasChild}^{I_2} &= \emptyset
\end{align*}
$$
Interpretations - Examples

TBox $\mathcal{T}$: \[\text{Man} \equiv \neg \text{Woman} \sqcap \text{Person}\]

$I_2 \models \mathcal{T}$: \[\text{Woman} \sqsubseteq \text{Person}\]

Mother $\equiv \text{Woman} \sqcap \exists \text{hasChild} \cdot \top$

ABox $\mathcal{A}$: \[\text{Man}(\text{STEPHEN}).\]

$I_2 \not\models \mathcal{A}$: \[\neg \text{Man}(\text{MONICA}).\]

Woman($\text{JESSICA}$).

hasManyChild($\text{STEPHEN}, \text{JESSICA}$).

\[\text{Man}^{I_2} = \{\text{STEPHEN}\}\]

\[\text{Woman}^{I_2} = \{\text{JESSICA}, \text{MONICA}\}\]

\[\text{Mother}^{I_2} = \emptyset\]

\[\text{Person}^{I_2} = \{\text{JESSICA}, \text{MONICA}, \text{STEPHEN}\}\]

\[\text{hasChild}^{I_2} = \emptyset\]
Interpretations - Examples

\[ \text{TBox } \mathcal{T} : \]
\[ \text{Man } \equiv \neg \text{Woman } \sqcap \text{Person} \]
\[ \text{Woman } \sqsubseteq \text{Person} \]
\[ \text{Mother } \equiv \text{Woman } \sqcap \exists \text{hasChild}. \top \]

\[ \text{ABox } \mathcal{A} : \]
\[ \text{Man}(\text{STEPHEN}). \]
\[ \neg \text{Man}(\text{MONICA}). \]
\[ \text{Woman}(\text{JESSICA}). \]
\[ \text{hasChild}(\text{STEPHEN}, \text{JESSICA}). \]

\[ \text{Man}^{I_3} = \{ \text{STEPHEN} \} \]
\[ \text{Woman}^{I_3} = \{ \text{JESSICA}, \text{MONICA} \} \]
\[ \text{Mother}^{I_3} = \{ \text{MONICA} \} \]
\[ \text{Person}^{I_3} = \{ \text{JESSICA}, \text{MONICA}, \text{STEPHEN} \} \]
\[ \text{hasChild}^{I_3} = \{ (\text{MONICA}, \text{STEPHEN}), (\text{STEPHEN}, \text{JESSICA}) \} \]
Interpretations - Examples

**TBox** $\mathcal{T}$:

\[
\text{Man} \equiv \neg \text{Woman} \sqcap \text{Person}
\]

$\mathcal{I}_3 \models \mathcal{T}$

\[
\text{Woman} \sqsubseteq \text{Person}
\]

\[
\text{Mother} \equiv \text{Woman} \sqcap \exists \text{hasChild}. \top
\]

**ABox** $\mathcal{A}$:

\[
\text{Man}(\text{STEPHEN}).
\]

$\mathcal{I}_3 \models \mathcal{A}$

\[
\neg \text{Man}(\text{MONICA}).
\]

\[
\text{Woman}(\text{JESSICA}).
\]

\[
\text{hasChild}(\text{STEPHEN}, \text{JESSICA}).
\]

\[
\text{Man}^{\mathcal{I}_3} = \{\text{STEPHEN}\}
\]

\[
\text{Woman}^{\mathcal{I}_3} = \{\text{JESSICA}, \text{MONICA}\}
\]

\[
\text{Mother}^{\mathcal{I}_3} = \{\text{MONICA}\}
\]

\[
\text{Person}^{\mathcal{I}_3} = \{\text{JESSICA}, \text{MONICA}, \text{STEPHEN}\}
\]

\[
\text{hasChild}^{\mathcal{I}_3} = \{(\text{MONICA}, \text{STEPHEN}), (\text{STEPHEN}, \text{JESSICA})\}
\]
Translation of TBox statements into Predicate Logic by using the mapping function $\pi$ (right side).

Let $C, D$ be complex classes, $R$ a role and $A$ an atomic class.

- $\pi(C \sqsubseteq D) = (\forall x(\pi_x(C) \to \pi_x(D)))$
- $\pi(C \equiv D) = (\forall x(\pi_x(C) \leftrightarrow \pi_x(D)))$
- $\pi_x(A) = A(x)$
- $\pi_x(\neg C) = \neg \pi_x(C)$
- $\pi_x(C \sqcap D) = \pi_x(C) \land \pi_x(D)$
- $\pi_x(C \sqcup D) = \pi_x(C) \lor \pi_x(D)$
- $\pi_x(\forall R.C) = (\forall y)(R(x, y) \to \pi_y(C))$
- $\pi_x(\exists R.C) = (\exists y)(R(x, y) \land \pi_y(C))$
- $\pi_y(A) = A(y)$
- $\pi_y(\neg C) = \neg \pi_y(C)$
- $\pi_y(C \sqcap D) = \pi_y(C) \land \pi_y(D)$
- $\pi_y(C \sqcup D) = \pi_y(C) \lor \pi_y(D)$
- $\pi_y(\forall R.C) = (\forall x)(R(y, x) \to \pi_x(C))$
- $\pi_y(\exists R.C) = (\exists x)(R(y, x) \land \pi_x(C))$
The following OWL DL language constructs are representable in $\mathcal{ALC}$:

- classes, roles, individuals
- class assertion, role assertion
- $\text{owl:Thing}$ und $\text{owl:Nothing}$
- class inclusion, equivalence and disjointness
- $\text{owl:intersectionOf}$, $\text{owl:unionOf}$
- $\text{owl:complementOf}$
- $\text{owl:allValuesFrom}$, $\text{owl:someValuesFrom}$
- $\text{rdfs:range}$ und $\text{rdfs:domain}$
Important inference problems

Global consistency of the knowledge base
- Does the knowledge base make sense?
  (Semantic: Exists a model for $\mathcal{K}$?)

Class consistency
- Has class $C$ to be empty?

Class inclusion (Subsumption)
- Structuring of the knowledge base

Class equivalence
- Are two classes the same?

Class disjointness
- Are two classes disjoint?

Class assertion
- Does individual $a$ belong to class $C$?

Instance generation (Retrieval) “find all $x$ with $C(x)$”
- Find all (known!) individuals of class $C$. 

$\mathcal{K} \models$ false?

$C \equiv \bot$?

$C \sqsubseteq D$?

$C \equiv D$?

$C \cap D = \bot$?

$C(a)$ ?
Outline

1. **OWL 2 Structure Overview**
2. **OWL 2 Semantics**
3. **Ontology Debugging and Repair via ORE**
4. **Ontology Enrichment via DL-Learner/ORE**
5. **Examples and Demo**
6. **Related Tools**
7. **Conclusions and Future Work**
Motivation

- increasing number of knowledge bases in the Semantic Web (see e.g. LOD cloud)
- maintenance of knowledge bases with expressive semantics is challenging

ORE simplifies this task (as part of the LOD2 stack)
Features of the ORE Tool

- open-source tool for repairing and extending knowledge bases
- state-of-the-art inconsistency detection, ranking, and repair methods
- use of supervised machine learning
- support for very large knowledge bases available as SPARQL endpoints
OWL ontology $\mathcal{O}$ consists of a set of axioms

capitalOf SubPropertyOf: locatedIn
BEIJING capitalOf CHINA
(note: Manchester OWL Syntax)
OWL ontology $\mathcal{O}$ consists of a set of axioms:

- capitalOf SubPropertyOf: locatedIn
- BEIJING capitalOf CHINA

(note: Manchester OWL Syntax)

The semantics of OWL allow to draw entailments:

- BEIJING locatedIn CHINA
• **OWL ontology** $\mathcal{O}$ consists of a set of axioms

  `capitalOf SubPropertyOf: locatedIn`
  `BEIJING capitalOf CHINA`

  (note: Manchester OWL Syntax)

• semantics of OWL allow to draw **entailments**

  `BEIJING locatedIn CHINA`

• **justification** $J \subseteq \mathcal{O}$ of an entailment is a minimal set of axioms from which the entailment can be drawn
$\mathcal{O}$ is inconsistent if it does not have a model (≡ contains a contradiction):

City and population some int[$>10000000$]
SubClassOf: capitalOf some Country
SHANGHAI Types: City,
not(capitalOf some Country)
Facts: population 13831900
a class $c$ in $\mathcal{O}$ is unsatisfiable if $C^I = \emptyset$ for all models $I$ of $\mathcal{O}$ (= the class cannot have an instance):

ISWCConference SubClassOf: SmallConference  
SmallConference SubClassOf:  
  not (hasEvent some (Tutorial or Workshop))  
ISWCConference SubClassOf:  
  (hasEvent some Tutorial) and  
  (hasEvent some Workshop)
Problems

- there can be many justifications for a single entailment
- there can be several unsatisfiable classes
- due to ontological relations, several problems can be intertwined and are difficult to separate
idea: separate between root and derived unsatisfiable classes

derived unsatisfiable class has justification, which contains a justification of another unsatisfiable class

fixing root problems may resolve further problems

idea: separate between root and derived unsatisfiable classes

derived unsatisfiable class has justification, which contains a justification of another unsatisfiable class

fixing root problems may resolve further problems

approach 1: compute all justifications for each unsatisfiable class and apply the definition → computationally often too expensive

approach 2: heuristics for structural analysis of axioms

ORE uses sound but incomplete variant of approach 2

resolving justification requires to delete or edit axioms

ranking methods highlight the most probable causes for problems

methods:

- frequency
- syntactic relevance
- semantic relevance
• resolving justification requires to delete or edit axioms
• ranking methods highlight the most probable causes for problems
• methods:
  • frequency
  • syntactic relevance
  • semantic relevance
• ORE supports those metrics and an aggregation of them

<table>
<thead>
<tr>
<th>Axiom</th>
<th>F</th>
<th>U</th>
<th>(\Sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumber \textit{SubClassOf} ForestProduct</td>
<td>2</td>
<td>15</td>
<td>0.88</td>
</tr>
<tr>
<td>Lumber \textit{SubClassOf} ManufacturedProduct</td>
<td>1</td>
<td>25</td>
<td>0.54</td>
</tr>
<tr>
<td>ForestProduct \textit{DisjointWith} ManufacturedProduct</td>
<td>2</td>
<td>35</td>
<td>0.71</td>
</tr>
</tbody>
</table>
- after repairing process, axioms have been deleted or modified
  → desired entailments may be lost or new entailments obtained (including inconsistencies!)
- ORE allows to preview new or lost entailments
  → user can decide to **preserve** them

<table>
<thead>
<tr>
<th>Retained</th>
<th>Olive</th>
<th>SubClassOf</th>
<th>OrganicObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lost</td>
<td>EdibleNut</td>
<td>DisjointWith</td>
<td>Fruit</td>
</tr>
</tbody>
</table>
ORE supports using **SPARQL endpoints**

- implements an **incremental load procedure**
- knowledge base is loaded in small chunks:
  - count number of axioms by type
  - **priority** based loading procedure
  - e.g. disjointness axioms have higher priority than class assertion axioms
- uses Pellet incremental reasoning

---

**“Learning of OWL Class Descriptions on Very Large Knowledge Bases”,**

- The algorithm performs **sanity checks**, e.g., SPARQL queries which probe for **typical inconsistent axiom sets**.
- It can fetch **additional Linked Data**.
- Different **termination criteria** are supported.
algorithm performs **sanity checks**, e.g. SPARQL queries which probe for **typical inconsistent axiom sets**

can fetch **additional Linked Data**

different termination criteria

**overall:**

- ORE allows to **apply state-of-the-art ontology debugging methods on a larger scale than was possible previously**
- aims at stronger support for the **“web aspect”** of the Semantic Web and the high popularity of Web of Data initiative
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
induction algorithms can derive axioms about a class by using its instances as positive examples
Induction algorithms can derive axioms about a class by using its instances as positive examples.

Data:

ISWC2003 Types: ISWCCConference
Facts: hasTopic SemanticBrokering,
hasTopic Ontologies
... 
takesPlaceIn Florida
induction algorithms can derive axioms about a class by using its instances as positive examples

Data:

ISWC2003 Types: ISWCCConference
Facts: hasTopic SemanticBrokering,
      hasTopic Ontologies
      ...
      takesPlaceIn Florida

Learned:

ISWCCconference SubClassOf: hasTopic some Ontologies
induction algorithms can derive axioms about a class by using its instances as positive examples

Data:

ISWC2003 Types: ISWCCConference
Facts: hasTopic SemanticBrokering,
      hasTopic Ontologies
      ...
      takesPlaceIn Florida

Learned:

ISWCCConference SubClassOf: hasTopic someOntologies

ISWCCConference SubClassOf: takesPlaceIn some
      (Europe or Asia or NorthAmerica)
DL-Learner is used as machine learning framework for enrichment
ORE uses DL-Learner (provides an ontology enrichment user interface on top of it)
ORE supports enriching an ontology with super class axioms and definitions (other algorithms will be supported in the future)
uses supervised CELOE machine learning algorithm implemented in DL-Learner
use class instances as positive examples

“DL-Learner: Learning Concepts in Description Logics”,

“Concept Learning in Description Logics Using Refinement Operators”,
J. Lehmann, P. Hitzler, Machine Learning journal, 2010
have to deal with very large knowledge bases → fragment extraction is used
often no perfectly accurate axioms in SW scenarios
ORE can handle consequences of adding such axioms

Example:

ORE/CELOE suggests that a “car” always has an “engine” and a “manufacturer”:
Class: Car SubClassOf: hasPart SOME Engine AND hasManufacturer SOME Company
but 3% of the cars do not have that information
→ ORE shows those instances and allows to complete information
What about other axioms besides complex super classes and definitions?
What about other axioms besides complex super classes and definitions?
It is possible to learn e.g. the property hierarchy, domains, ranges etc?
How can this be done efficiently?
General method:

1. Use SPARQL queries to obtain general/schema information about the knowledge base, in particular we retrieve axioms, which allow to construct the class hierarchy.

2. Obtain data via SPARQL, which is relevant for the learning the considered axiom.

3. Compute appropriate score of axiom candidates and return results.
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .

dbpedia: Luxembourg dbo: currency dbpedia: Euro ;
    rdf:type dbo: Country .
dbpedia: Ecuador dbo: currency dbpedia: United_States_dollar ;
    rdf:type dbo: Country .
dbpedia: Ifni dbo: currency dbpedia: Spanish_peseta ;
    rdf:type dbo: PopulatedPlace .
dbo: Country rdfs: subClassOf dbo: PopulatedPlace .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
dbpedia: Luxembourg dbo: currency dbpedia: Euro ;
    rdf: type dbo: Country .
dbpedia: Ecuador dbo: currency dbpedia: United_States_dollar ;
    rdf: type dbo: Country .
dbpedia: Ifni dbo: currency dbpedia: Spanish_peseta ;
    rdf: type dbo: PopulatedPlace .
dbo: Country rdfs: subClassOf dbo: PopulatedPlace .

**Query in Phase 2**

```sparql
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?type COUNT(DISTINCT ?ind) WHERE {
  ?ind dbo: currency ?o .
  ?ind a ?type .
} GROUP BY ?type
```
Example: Domain of object property `dbo:currency`

Query

```sparql
PREFIX dbo: <http://dbpedia.org/ontology/>
PREIFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
SELECT ?type COUNT(DISTINCT ?ind) WHERE {
  ?ind dbo:currency ?o.
  ?ind a ?type.
} GROUP BY ?type

@prefix dbo: <http://dbpedia.org/ontology/> .
dbpedia: Luxembourg dbo: currency dbpedia: Euro .
dbpedia: Ecuador dbo: currency dbpedia: United_States_dollar .
db: PopulatedPlace .
dbo: Country rdfs: subClassOf dbo: PopulatedPlace .
```

- Score(`dbo:Country`) = 2/3 = 66.7%
- Score(`dbo:PopulatedPlace`) = 3/3 = 100%
  (33.3% without inference)
Example: Domain of object property `dbo:currency`

```query
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?type COUNT(DISTINCT ?ind) WHERE {
    ?ind dbo:currency ?o.
    ?ind a ?type.
} GROUP BY ?type
```

- Score(`dbo:Country`) = 2/3 = 66.7%
- Score(`dbo:PopulatedPlace`) = 3/3 = 100%  
  
  (33.3% without inference)

Problem: Straightforward score doesn’t take the **support** for an axiom in the knowledge base into account!
Score method: Average of the 95% confidence interval boundaries (can be computed efficiently using improved Wald method)
Score method: Average of the 95% confidence interval boundaries (can be computed efficiently using improved Wald method)

**95% confidence interval (Wald method)**

Assume we have \( m \) observations out of which \( s \) were successful, then the approximation of the 95% confidence interval is as follows:

\[
\max(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1 - p')}{m + 4}}) \text{ to } \min(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1 - p')}{m + 4}})
\]

with \( p' = \frac{s + 2}{m + 4} \)
Score method: Average of the 95% confidence interval boundaries (can be computed efficiently using improved Wald method)

95% confidence interval (Wald method)

Assume we have \( m \) observations out of which \( s \) were successful, then the approximation of the 95% confidence interval is as follows:

\[
\max(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1 - p')}{m + 4}}) \quad \text{to} \quad \min(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1 - p')}{m + 4}})
\]

with \( p' = \frac{s + 2}{m + 4} \)

- toy example: score(dbo:Country) = 57.3% (2 of 3)
- toy example: score(dbo:PopulatedPlace) = 69.1% (3 of 3)
Score method: Average of the 95% confidence interval boundaries (can be computed efficiently using improved Wald method)

95% confidence interval (Wald method)

Assume we have $m$ observations out of which $s$ were successful, then the approximation of the 95% confidence interval is as follows:

$$
\text{max}(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}}) \quad \text{to} \quad \text{min}(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}})
$$

with $p' = \frac{s + 2}{m + 4}$

- toy example: score(dbo:Country) = 57.3% (2 of 3)
- toy example: score(dbo:PopulatedPlace) = 69.1% (3 of 3)
- DBpedia Live: score(dbo:Country) = 98.5% (603 of 610)
- DBpedia Live: score(dbo:PopulatedPlace) = 97.1% (594 of 610)
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
To install DL-Learner, perform the following steps:

- install Java version 6 or higher
  (http://www.java.com/en/download/)
- go to http://dl-learner.org and press “download”
- extract the downloaded archive
- run cli/enrichment -? (Unix) or cli/enrichment.bat -? (Windows) on the command line in the extracted directory
Option                                Description
-----                                -----------
-?, -h, --help                        Show help.
-e, --endpoint <URL>                  SPARQL endpoint URL to be used.
-f, --format                          Format of the generated output (plain,
                                       rdf/xml, turtle, n-triples).
                                       (default: plain)
-g, --graph [URI]                     URI of default graph for queries on
                                       SPARQL endpoint.
-i, --inference [Boolean]             Specifies whether to use inference. If
                                       yes, the schema will be loaded into
                                       a reasoner and used for computing
                                       the scores. (default: true)
-o, --output [File]                   Specify a file where the output can be
                                       written.
-r, --resource [URI]                  The resource for which enrichment
                                       axioms should be suggested.
-t, --threshold [Double]              Confidence threshold for suggestions.
                                       Set it to a value between 0 and 1.
                                       (default: 0.7)
Obtain enrichment suggestions for the currency property in DBpedia Live:

```
-e http://live.dbpedia.org/sparql -g http://dbpedia.org
-r http://dbpedia.org/ontology/currency
```

Output those enrichments to a file `results.txt`:

```
-e http://live.dbpedia.org/sparql -g http://dbpedia.org
-r http://dbpedia.org/ontology/currency -o results.txt
```

Write the enrichments in Turtle syntax in a file using the enrichment ontology:

```
-e http://live.dbpedia.org/sparql -g http://dbpedia.org
-r http://dbpedia.org/ontology/currency -o results.ttl
-f turtle
```

Do the same task with an increased threshold and without inference

```
-e http://live.dbpedia.org/sparql -g http://dbpedia.org
-r http://dbpedia.org/ontology/currency -o results.ttl
-f turtle -t 0.9 -i false
```
Applying disjoint objectproperty axiom learner on http://dbpedia.org/ontology/currency ... done in 274 ms
suggested axioms and their score in percent:
100,0% dbo:currency DisjointWith: dbo:academicAdvisor
100,0% dbo:currency DisjointWith: dbo:academicAward
100,0% dbo:currency DisjointWith: dbo:actingHeadteacher
100,0% dbo:currency DisjointWith: dbo:administrativeCollectivity
100,0% dbo:currency DisjointWith: dbo:administrativeDistrict
100,0% dbo:currency DisjointWith: dbo:administrator
100,0% dbo:currency DisjointWith: dbo:affiliation
100,0% dbo:currency DisjointWith: dbo:afiAward
100,0% dbo:currency DisjointWith: dbo:aircraftAttack
100,0% dbo:currency DisjointWith: dbo:aircraftBomber

Applying equivalent objectproperty axiom learner on http://dbpedia.org/ontology/currency ... done in 56 ms
no axiom suggested

Applying functional objectproperty axiom learner on http://dbpedia.org/ontology/currency ... done in 34 ms
suggested axioms and their score in percent:
75,9% Functional(dbo:currency)

Applying inversefunctional objectproperty axiom learner on http://dbpedia.org/ontology/currency ... done in 31 ms
no axiom suggested

Applying objectproperty domain axiom learner on http://dbpedia.org/ontology/currency ... done in 1603 ms
suggested axioms and their score in percent:
98,5% Domain(dbo:currency, dbo:PopulatedPlace)
98,5% Domain(dbo:currency, dbo:Place)
97,1% Domain(dbo:currency, dbo:Country)

Applying objectproperty range learner on http://dbpedia.org/ontology/currency ... done in 183 ms
no axiom suggested

Applying object subPropertyOf axiom learner on http://dbpedia.org/ontology/currency ... done in 53 ms
go to http://web.ore-tool.net [Warning: alpha version!]
go to http://web.ore-tool.net [Warning: alpha version!]
(1) click on Bookmark >>> Koala and Action >>> Debug
have a look at the generated justifications
- go to http://web.ore-tool.net [Warning: alpha version!]
- (1) click on Bookmark >> Koala and Action >> Debug
- have a look at the generated justifications
- (2) click on Bookmark >> Swore and Action >> Learn
- click on CustomerRequirement and have a look at the learned definitions
ORE - Demo

- go to http://web.ore-tool.net [Warning: alpha version!]
- (1) click on Bookmark >> Koala and Action >> Debug
- have a look at the generated justifications
- (2) click on Bookmark >> Swore and Action >> Learn
- click on CustomerRequirement and have a look at the learned definitions
- (3) click on File >> Connect to SPARQL Endpoint
- try http://live.dbpedia.org/sparql with default graph http://dbpedia.org
- note: only CELOE integrated in ORE at the moment, many algorithms on our TODO list ...
Inconsistency in DBpedia Live:

Individual: dbr:Purify_(album)
  Facts: dbo:artist dbr:Axis_of_Advance
Individual: dbr:Axis_of_Advance
  Types: dbo:Organisation
Class: dbo:Organisation
  DisjointWith dbo:Person
ObjectProperty: dbo:artist
  Range: dbo:Person
Inconsistency in DBpedia in combination with WGS84 (Linked Data):

Types: dbo:Organisation
DataProperty: geo:long  Domain: geo:SpatialThing
Class: dbo:Organisation  DisjointWith: geo:SpatialThing
Inconsistency in OpenCyc:

Individual: 'PopulatedPlace'
  Types: 'ArtifactualFeatureType', 'ExistingStuffType'
Class: 'ArtifactualFeatureType'
  SubClassOf: 'ExistingObjectType'
Class: 'ExistingObjectType'
  DisjointWith: 'ExistingStuffType'
## Explanations

<table>
<thead>
<tr>
<th>Explanation type</th>
<th>Explanation count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show regular explanations</td>
<td>2</td>
</tr>
<tr>
<td>Show laconic explanations</td>
<td>8</td>
</tr>
<tr>
<td>Limit explanation count to:</td>
<td>2</td>
</tr>
</tbody>
</table>

### Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>F</th>
<th>U</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>KoalaWithPhD EquivalentTo Koala and (hasDegree value PhD)</td>
<td>2</td>
<td>8</td>
<td>0.39</td>
</tr>
<tr>
<td>hasDegree Domain Person</td>
<td>1</td>
<td>8</td>
<td>0.22</td>
</tr>
<tr>
<td>Koala SubClassOf Marsupials</td>
<td>3</td>
<td>8</td>
<td>0.56</td>
</tr>
<tr>
<td>Marsupials DisjointWith Person</td>
<td>3</td>
<td>8</td>
<td>0.56</td>
</tr>
</tbody>
</table>

### Repair plan

<table>
<thead>
<tr>
<th>Repair plan</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasDegree Domain Person</td>
<td>Lost hasDegree Domain Animal</td>
</tr>
</tbody>
</table>

Execute
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
**Related Tools**

- **Swoop**
  - can compute justifications for unsatisfiability of classes and offers repair mode
  - fine-grained justification computation algorithm is incomplete
  - can also compute justifications for an inconsistent ontology, but does not offer repair mode in this case
  - does not extract locality-based modules, which leads to lower performance for large ontologies

- **RaDON**
  - plugin for the NeOn toolkit
  - offers a number of techniques for working with inconsistent or incoherent ontologies
  - allows to reason with inconsistent ontologies and can handle sets of ontologies (ontology networks)
  - no fine-grained justifications, no repair impact analysis

- **Pellint**
  - searches for common patterns which lead to potential reasoning performance problems
  - integration in ORE planned
**Related Tools II**

- **PION and DION**
  - developed in the SEKT project to deal with inconsistencies
  - PION is an inconsistency tolerant reasoner (four-valued paraconsistent logic)
  - DION offers the possibility to compute justifications, but no repair

- **Explanation Workbench**
  - Protégé plugin for reasoner requests like class unsatisfiability or inferred subsumption relations
  - can compute regular and laconic justifications
  - motivated the ORE debugging interface
  - current version of Explanation Workbench does not allow to remove axioms in laconic justifications

- **RepairTab**
  - supports the user in finding and detecting errors in ontologies
  - RepairTab uses a modified tableau algorithm
  - shows inferences which can no longer be drawn after removing an axiom (inspired ORE)
Outline

1. OWL 2 Structure Overview
2. OWL 2 Semantics
3. Ontology Debugging and Repair via ORE
4. Ontology Enrichment via DL-Learner/ORE
5. Examples and Demo
6. Related Tools
7. Conclusions and Future Work
ORE and DL-Learner development continues within the LOD2 project.
DL-Learner with enrichment support released two weeks ago (alpha).
ORE 0.2 release is a desktop application (mostly for local OWL files).
Next release will include a web version (live prototype on web.ore-tool-net).
Support for detection of further modeling problems.
Improving Linked Data / SPARQL component - debugging LOD knowledge bases will be a major use case.
Constant evolution/extension of underlying methods, e.g., automatic lemma generation, proofs, textual justifications.
Conclusions

- OWL based on model theoretic semantics
- ORE and DL-Learner are open source tools for ontology repair and enrichment
- support re-active ontology engineering (see ISSLOD talk by Vojtech Svatek)
- ORE uses state-of-the-art ontology debugging and learning methods
- combines advantages of different existing tools and methods
- will be able to detect modeling problems in very large knowledge bases
- long term goal: build a bridge between the current “Web of Data” and expressive OWL semantics
ORE

http://ore-tool.net

AKSW/MOLE Group, University of Leipzig