Textbook (required)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)

http://www.semantic-web-book.org
Today: OWL Syntax
OWL/RDF Semantic Mismatch

\[ \text{ex:speaksWith} \ rdfs:domain \ \text{ex:Homo} . \]
\[ \text{ex:Homo} \ rdfs:subClassOf \ \text{ex:Primates} . \]

\textbf{does not RDFS-entail}

\textbf{although it is a valid OWL entailment.}

\textbf{It does RDFS-entail}

\[ \text{rdfs:subClassOf} \ \text{rdf:type} \ \text{rdf:Property} \]

\textbf{which is not a valid OWL entailment.}
OWL 2 Syntaxes

• RDF/XML Syntax
  – The only *normative* syntax (i.e. to be OWL 2 compliant, a tool has to support this (and only this) syntax.

• Turtle Syntax
  – Straightforward Turtle version of the RDF/XML Syntax.
  – We will cover the RDF Syntax using Turtle or RDF/XML.

• Functional Style Syntax
  – Prefix-syntax, given as formal grammar
  – Clean, adjustable, modifiable, easily parsable
  – Used for *defining* OWL 2 in the W3C Specs.

• Manchester Syntax
  – User-friendly(?) syntax, used e.g. in Protégé 4

• OWL/XML Syntax
  – Notational variant of the Functional Style Syntax.
  – Does not use RDF triples, but simply XML tree structure.
Syntax Examples

• Many examples, translated into all syntaxes:

• Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph
OWL 2 Web Ontology Language: Primer.
W3C Recommendation, 27 October 2009.
http://www.w3.org/TR/owl2-primer/
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
The RDFS perspective

- :mary rdf:type :Person.
- :Mother rdfs:subClassOf :Woman.
- :john :hasWife :Mary.
- :hasWife rdfs:subPropertyOf :hasSpouse.

- :hasWife rdfs:range :Woman.
- :hasWife rdfs:domain :Man.

- owl:Thing
- owl:Nothing
- owl:topProperty
- owl:bottomProperty

owl namespace: http://www.w3.org/2002/07/owl#
SROIQ(D) constructors – overview

- ABox assignments of individuals to classes or properties
- ALC: $\subseteq, \equiv$ for classes
  \[ \cap, \cup, \neg, \exists, \forall \]
  \[ \top, \bot \]
- SR: + property chains, property characteristics, role hierarchies $\subseteq$
- SRO: + nominals \{o\}
- SROI: + inverse properties
- SROIQ: + qualified cardinality constraints
- SROIQ(D): + datatypes (including facets)
  + top and bottom roles (for objects and datatypes)
  + disjoint properties
  + Self
  + Keys (not in SROIQ(D), but in OWL)
RDF Syntax Challenges

• How do you put SROIQ(D) axioms like

\[
\text{Orphan} \sqsubseteq \text{Human} \sqcap \forall \text{hasParent.} \neg \text{Alive}
\]

into a graph structure?

• How do you do it such that the RDF Schema semantics and the DL semantics are not violated?

• How do you do it without violating the main conceptual ideas behind RDF and DLs?

• That’s actually impossible without violating either RDF or DL. We have to do some approximations, and accept that the layering cannot be perfect.
General Idea

Orphan \sqsubseteq \text{Human} \sqcap \forall \text{hasParent}. \neg \text{Alive}
RDF Syntax

- From here on, you can basically make the RDF syntax yourself.

- You only need to know the OWL vocabulary to use and some constructs need some design decisions, which are sometimes almost arbitrary.
RDF Semantics?

- You get all kinds of entailments which are entirely irrelevant for the OWL knowledge base.

- `owl:complementOf rdf:type rdf:Property .`
- `_x5 owl:complementOf _:xyz .`
- `owl:Restriction rdf:type rdfs:Class .`
- `_:hasParent rdf:type rdfs:Resource .`
- `owl:Restriction rdfs:subClassOf rdfs:Resource .`
- `owl:Restriction rdfs:subClassOf owl:Restriction .`
OWL RDF Syntax: Individuals

:Mary rdf:type :Woman .
:John :hasWife :Mary .
:John owl:differentFrom :Bill .
:James owl:sameAs :Jim .
:John :hasAge "51"^^xsd:nonNegativeInteger .

\[ \{\text{John}\} \cap \{\text{Bill}\} \subseteq \perp \]

\[ \{\text{John}\} \equiv \{\text{Jim}\} \]

\[ \neg \text{hasWife(Bill,Mary)} \]

[[] rdf:type owl:NegativePropertyAssertion ;
owl:sourceIndividual :Bill ;
owl:assertionProperty :hasWife ;
owl:targetIndividual :Mary .

[[] rdf:type owl:NegativePropertyAssertion ;
owl:sourceIndividual :Jack ;
owl:assertionProperty :hasAge ;
owl:targetValue 53 .

KR4SW – Winter 2012 – Pascal Hitzler
OWL RDF Syntax: Classes + Properties

:Woman rdfs:subClassOf :Person .

:Person owl:equivalentClass :Human .

\[
\begin{array}{c}
\text{Woman} \\
\text{Man}
\end{array}
\]

:hasWife rdfs:subPropertyOf :hasSpouse .

:hasWife rdfs:domain :Man ;
  rdfs:range :Woman .
OWL RDF Syntax: Complex Classes

Mother owl:equivalentClass [  
  rdf:type owl:Class;  
  owl:intersectionOf ( :Woman :Parent )  
].

Parent owl:equivalentClass [  
  rdf:type owl:Class;  
  owl:unionOf ( :Mother :Father )  
].

ChildlessPerson owl:equivalentClass [  
  rdfs:subClassOf owl:Class;  
].

\[
\text{Mother} \equiv \text{Woman} \sqcap \text{Parent}\\
\text{Parent} \equiv \text{Mother} \sqcup \text{Father}\\
\text{ChildlessPerson} \equiv \text{Person} \sqcap \neg \text{Parent}
\]
OWL RDF Syntax: Complex Classes

```
```

Person ⋂ ¬Parent (Jack)
OWL RDF Syntax: Restrictions

```
:Parent  owl:equivalentClass [  
    rdf:type            owl:Restriction ;  
    owl:onProperty     :hasChild ;  
    owl:someValuesFrom :Person
] .
```

Parent $\equiv \exists \text{hasChild}. \text{Person}$

```
:Orphan  owl:equivalentClass [  
    rdf:type                owl:Restriction ;  
    owl:onProperty          [ owl:inverseOf :hasChild ] ;  
    owl:allValuesFrom       :Dead
] .
```

Orphan $\equiv \forall \text{hasChild}^\sim. \text{Dead}$
OWL RDF Syntax: Restrictions

:JohnsChildren owl:equivalentClass [ rdf:type owl:Restriction ; owl:onProperty :hasParent ; owl:hasValue :John ] .

\[ :\text{JohnsChildren} \equiv \exists \text{hasParent}.\{\text{John}\} \]

: NarcisticPerson owl:equivalentClass [ rdf:type owl:Restriction ; owl:onProperty :loves ; owl:hasSelf "true"^^xsd:boolean .

\[ :\text{NarcisticPerson} \equiv \exists \text{loves}.\text{Self} \]
OWL RDF Syntax: Restrictions

:John rdf:type [ rdf:type owl:maxQualifiedCardinality owl:onProperty owl:onClass  ≤4 hasChild.Parent (John)  ].

:John rdf:type [ rdf:type owl:minQualifiedCardinality owl:onProperty owl:onClass  ≥2 hasChild.Parent (John)  ].

:John rdf:type [ owl:Restriction ; owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ; :hasChild ; :Parent  =3 hasChild.Parent (John)  ].

:John rdf:type [ owl:Restriction ; owl:minQualifiedCardinality "3"^^xsd:nonNegativeInteger ; :hasChild ; :Parent  ].
OWL RDF Syntax: Restrictions

```rml
:John rdf:type [ 
    rdf:type owl:Restriction ;
    owl:cardinality "5"^^xsd:nonNegativeInteger ;
    owl:onProperty :hasChild ] .
```

```
=5 hasChild .T (John)
```

```rml
:MyBirthdayGuests owl:equivalentClass [ 
    rdf:type owl:Class ;
    owl:oneOf ( :Bill :John :Mary ) ] .
```

```
MyBirthdayGuests ≡ {Bill, John, Mary}
```
\textbf{OWL RDF Syntax: Properties}

\begin{verbatim}
:hasParent owl:inverseOf :hasChild .

Orphan owl:equivalentClass [ rdf:type owl:Restriction ;
owl:onProperty [ owl:complementOf :hasChild ] ;
owl:allValuesFrom :Dead ] .

:hasSpouse rdf:type owl:SymmetricProperty .
:hasChild rdf:type owl:AsymmetricProperty .
:hasParent owl:propertyDisjointWith :hasSpouse .
:hasRelative rdf:type owl:ReflexiveProperty .
:parentOf rdf:type owl:IrreflexiveProperty .
:hasHusband rdf:type owl:InverseFunctionalProperty .
:hasAncestor rdf:type owl:TransitiveProperty .
\end{verbatim}

\textbf{Orphan} $\equiv \forall \text{hasChild} . \text{Dead}$
In OWL 2 a collection of (data or object) properties can be assigned as a key to a class expression. This means that each named instance of the class expression is uniquely identified by the set of values which these properties attain in relation to the instance.
owl:Datatype
owl:withFacets (  
    [ xsd:minInclusive "0"^^xsd:integer ]
    [ xsd:maxInclusive "150"^^xsd:integer ]
)
].

:minorAge owl:equivalentClass
[ rdf:type  rdfs:Datatype;
  owl:intersectionOf (  
    :personAge
    [ rdf:type  rdfs:Datatype;
      owl:datatypeComplementOf  :minorAge ]
  )
].

:toddlerAge owl:equivalentClass
[ rdf:type  rdfs:Datatype;
  owl:oneOf ( "1"^^xsd:integer  "2"^^xsd:integer )
].
## Essential OWL Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Related OWL vocabulary</th>
<th>FOL</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>top/bottom class</td>
<td><code>owl:Thing/owl:Nothing</code></td>
<td>(axiomatise)</td>
<td>$T/\bot$</td>
</tr>
<tr>
<td>Class intersection</td>
<td><code>owl:intersectionOf</code></td>
<td>$\land$</td>
<td>$\cap$</td>
</tr>
<tr>
<td>Class union</td>
<td><code>owl:unionOf</code></td>
<td>$\lor$</td>
<td>$\cup$</td>
</tr>
<tr>
<td>Class complement</td>
<td><code>owl:complementOf</code></td>
<td>$\neg$</td>
<td>$\neg$</td>
</tr>
<tr>
<td>Enumerated class</td>
<td><code>owl:oneOf</code></td>
<td>(ax. with $\approx$)</td>
<td>${a}$</td>
</tr>
<tr>
<td><strong>Property restrictions</strong></td>
<td><code>owl:onProperty</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existential</td>
<td><code>owl:someValueFrom</code></td>
<td>$\exists y \ldots$</td>
<td>$\exists R.C$</td>
</tr>
<tr>
<td>Universal</td>
<td><code>owl:allValuesFrom</code></td>
<td>$\forall y \ldots$</td>
<td>$\forall R.C$</td>
</tr>
<tr>
<td>Min. cardinality</td>
<td><code>owl:minQualifiedCardinality</code></td>
<td>$\exists y_1 \ldots y_n \ldots$</td>
<td>$\geq n \text{ S.C}$</td>
</tr>
<tr>
<td></td>
<td><code>owl:onClass</code></td>
<td>$\forall y_1 \ldots y_n + 1. \ldots \rightarrow \ldots$</td>
<td>$\leq n \text{ S.C}$</td>
</tr>
<tr>
<td>Max. cardinality</td>
<td><code>owl:maxQualifiedCardinality</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local reflexivity</td>
<td><code>owl:hasSelf</code></td>
<td>$R(x,x)$</td>
<td>$\exists R.Self$</td>
</tr>
</tbody>
</table>
## Essential OWL Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Related OWL vocabulary</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property chain</td>
<td>owl:propertyChainAxiom</td>
<td>$\circ$</td>
</tr>
<tr>
<td>Inverse</td>
<td>owl:inverseOf</td>
<td>$R^\rightarrow$</td>
</tr>
<tr>
<td>Key</td>
<td>owl:hasKey</td>
<td></td>
</tr>
<tr>
<td>Property disjointness</td>
<td>owl:propertyDisjointWith</td>
<td>Dis(R,S)</td>
</tr>
</tbody>
</table>

### Property characteristics

<table>
<thead>
<tr>
<th>Symmetric</th>
<th>owl:SymmetricProperty</th>
<th>$\text{Sym}(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric</td>
<td>owl:AsymmetricProperty</td>
<td>$\text{Asy}(R)$</td>
</tr>
<tr>
<td>Reflexive</td>
<td>owl:ReflexiveProperty</td>
<td>$\text{Ref}(R)$</td>
</tr>
<tr>
<td>Irreflexive</td>
<td>owl:IrreflexiveProperty</td>
<td>$\text{Irr}(R)$</td>
</tr>
<tr>
<td>Transitivity</td>
<td>owl:TransitiveProperty</td>
<td>$\text{Tra}(R)$</td>
</tr>
</tbody>
</table>

### Subclass

<table>
<thead>
<tr>
<th>Subclass</th>
<th>rdfs:subClassOf</th>
<th>$\forall x. C(x) \rightarrow D(x)$</th>
<th>$\text{C} \sqsubseteq \text{D}$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Subproperty</th>
<th>rdfs:subPropertyOf</th>
<th>$\forall x,y. R(x,y) \rightarrow S(x,y)$</th>
<th>$\text{R} \sqsubseteq \text{S}$</th>
</tr>
</thead>
</table>
OWL RDF Syntax: Header

@prefix : <http://example.com/owl/families/> .
@prefix otherOnt: <http://example.org/otherOntologies/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.com/owl/families>
owl:imports <http://example.org/otherOntologies/families/> .
@prefix : <http://example.com/owl/families/> .
@prefix otherOnt: <http://example.org/otherOntologies/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


:Mary owl:sameAs otherOnt:MaryBrown .
:John owl:sameAs otherOnt:JohnBrown .
:Adult owl:equivalentClass otherOnt:Grownup .
:hasChild owl:equivalentProperty otherOnt:child .
:hasAge owl:equivalentProperty otherOnt:age .
Each class, property, or individual needs to be declared.

Punning:

Same URI can stand e.g. for both an individual and a class:

\[ :\text{John} \ \text{rdf\:type} \ \text{owl\:NamedIndividual} . \]
\[ :\text{Person} \ \text{rdf\:type} \ \text{owl\:Class} . \]
\[ :\text{hasWife} \ \text{rdf\:type} \ \text{owl\:ObjectProperty} . \]
\[ :\text{hasAge} \ \text{rdf\:type} \ \text{owl\:DatatypeProperty} . \]

Semantics: This is semantically interpreted as if the two occurrences of Father were in fact distinct.

Not allowed: E.g. use of a URI for both object and datatype property.
:Person rdfs:comment "Represents the set of all people."^^xsd:string .

[ ] rdf:type owl:Axiom ;
owl:annotatedSource :Man ;
owl:annotatedProperty rdfs:subClassOf ;
rdfs:comment "States that every man is a person."^^xsd:string .
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
OWL DL and OWL Full

• OWL 2 DL is the “description logic” version of OWL
  – global restrictions from SROIQ(D) apply
  – RDF can only be used in a very controlled fashion (only what is necessary for expressing OWL axioms)
  – model-theoretic semantics of SROIQ(D) is used, called OWL 2 Direct Semantics

• OWL 2 Full is unrestricted OWL 2 DL plus all of RDF(S).
  – no global restrictions
  – RDF can be used freely
  – semantics is a hybrid of RDFS and OWL 2 DL semantics, called RDF-Based Semantics

• Both semantics are in the W3C recommendation. No implementations of the OWL 2 Full semantics exist.
OWL 2 Profiles

• The OWL 2 spec describes three profiles (fragments, sublanguages) which have polynomial complexity.
  – OWL EL (the description logic EL++)
  – OWL QL (the description logic DL LiteR)
  – OWL RL (the description logic DLP)
    • inspired by intersecting OWL with Datalog
    • implemented e.g. in Oracle 11g
OWL Functional Syntax

```
SubClassOf(:ChildlessPerson
  ObjectIntersectionOf(
    :Person
  ObjectComplementOf(
    ObjectInverseOf(:hasParent)
    owl:Thing
  )
)
)

ChildlessPerson ⊑ Person ⊓ ∃hasParent¬.⊤
```

```
ClassAssertion(
  ObjectIntersectionOf(
    :Person
  ObjectComplementOf(:Parent)
  )
  :Jack
)

Person ⊓ ¬Parent(Jack)
```
Individual: John
Types: Father
Types: hasChild max 4 Parent
Types: hasChild min 2 Parent
Types: hasChild exactly 3 Parent
Types: hasChild exactly 5
Facts: hasAge "51"^^xsd:integer
Facts: hasWife Mary
DifferentFrom: Bill
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
Class project: next step

• Make sure your ontology is actually in OWL DL (i.e. remove or remodel things which are not in OWL DL).

• Put your OWL ontology into Protégé: 
  
  http://protege.stanford.edu/

• Send the ontology serialized by Protégé (in RDF/XML or Turtle) to me by 22nd of February.
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
Presentation format

• 25 minutes (timing will be strict) plus a few questions

• Content selection is up to you. Presentation must contain the key results from the paper you are presenting. It can also contain material related to these key results which are not in the paper itself (but which you may find more interesting than the rest of the paper).
Presenation evaluation dimensions

Example criteria – may not all be weighted equally:

• Quality of slides
• Quality and effectiveness of explanations
• Quality of presentation style (use of verbal and body language, use of media, flexibility in case of interaction with audience, time management)
• Correctness of content
• Grade of reaching the audience and getting the content across
• How “interesting” the material is presented
• Competence in answering questions
Tuesday 10\textsuperscript{th} of January: RDF Schema
Thursday 12\textsuperscript{th} of January: RDF and RDFS Semantics
Tuesday 17\textsuperscript{th} of January: RDF and RDFS Semantics
Thursday 19\textsuperscript{th} of January: exercise session 1
Tuesday 24\textsuperscript{th} of January: OWL part 1 – Description Logics
Thursday 2\textsuperscript{nd} of February: OWL pt 2 – model-theoretic Semantics
Tuesday 7\textsuperscript{th} of February: Partonomies
Thursday 9\textsuperscript{th} of February: SPARQL
Tuesday 14\textsuperscript{th} of February: OWL part 3 – web syntax
Thursday 16\textsuperscript{th} of February: exercise session 2
Tuesday 21\textsuperscript{st} of February: OWL part 4 – Tableaux calculus
Thursday 23\textsuperscript{rd} of February: Rules
Tuesday 28\textsuperscript{th} of February: exercise session 3