OWL Syntax and Relation to RDF

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
Textbook

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
Textbook – Chinese translation

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

语义Web技术基础
Tsinghua University Press (清华大学出版社)，2013.

Translators:
Yong Yu, Haofeng Wang, Guilin Qi (俞勇，王昊奋，漆桂林)

http://www.semantic-web-book.org
Semantic Web journal

- EiCs: Pascal Hitzler
  Krzysztof Janowicz

- New journal with significant initial uptake.

- We very much welcome contributions at the “rim” of traditional Semantic Web research – e.g., work which is strongly inspired by a different field.

- Non-standard (open & transparent) review process.

- [http://www.semantic-web-journal.net/](http://www.semantic-web-journal.net/)
The Kno.e.sis Center and My Lab

- Ohio Center of Excellence in Knowledge-enabled Computing Director: Amit Sheth
- 15 faculty (8 in Computer Science) across 4 Departments, with ca. 50 PhD students

- Knowledge Engineering Lab (since January 2010) Led by myself

Currently
- 8 PhD students
- 2 Master students
- 3 undergrads

- http://www.knoesis.org/
In World Wide Web research
out of 2,507 organizations world-wide,
Wright State ranked 6\textsuperscript{th} in January 2013
The Semantic Web Stack

- User Interface & applications
- Trust
  - Proof
  - Unifying Logic
    - Query: SPARQL
    - ontology: OWL
    - Rules: RIF
    - RDF-S
  - Data interchange: RDF
    - XML
    - URI
    - Unicode
- Crypto
Contents:

- Overview
- RDF Syntax in Detail
- Other Syntaxes and OWL Variants
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
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: \( \sqsubseteq, \equiv \) for classes
  \( \cap, \cup, \neg, \exists, \forall \)
  \( \top, \bot \)
- SR: + property chains, property characteristics, role hierarchies \( \sqsubseteq \)
- 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.
Contents:

• Overview
• RDF Syntax in Detail
• Other Syntaxes and OWL Variants
General Idea

Orphan $\sqsubseteq$ Human $\sqcap \forall$ hasParent. $\neg$ Alive

Diagram:

- Orphan
- Human
- Alive
- hasParent
- owl:Restriction

Arrows:
- rdf:first
- rdf:rest
- owl:onProperty
- rdf:type
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 .

{John} \cap \{Bill\} \subseteq \bot
{John} \equiv \{Jim\}

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

\neg hasWife(Bill,Mary)

[] rdf:type owl:NegativePropertyAssertion ;
  owl:sourceIndividual :Jack ;
  owl:assertionProperty :hasAge ;
  owl:targetValue 53 .
OWL RDF Syntax: Classes + Properties

:Woman rdfs:subClassOf :Person .

:Person owl:equivalentClass :Human .

[ ] rdf:type owl:AllDisjointClasses ;
owl:members ( :Woman :Man ) .

\[ \text{Woman} \sqcap \text{Man} \sqsubseteq \bot \]

: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 ) ].


:Grandfather rdfs:subClassOf [ rdf:type owl:Class ; owl:intersectionOf ( :Man :Parent ) ].

Mother ≡ Woman □ Parent

Parent ≡ Mother □ Father

ChildlessPerson ≡ Person □ ¬Parent
Person $\sqcap \neg$Parent (Jack)

```owl
```
OWL RDF Syntax: Restrictions

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

\[ \text{Parent} \equiv \exists \text{hasChild} . \text{Person} \]

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

\[ \text{Orphan} \equiv \forall \text{hasChild} \neg . \text{Dead} \]
OWL RDF Syntax: Restrictions

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

JohnsChildren ≡ ∃hasParent.{John}
```

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

NarcisticPerson ≡ ∃loves.Self
```
OWL RDF Syntax: Restrictions

\[
\leq 4 \text{ hasChild.Parent (John)}
\]

```
:John rdf:type [  
  rdf:type owl:Restriction ;
  owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ;
  owl:onProperty :hasChild ;
  owl:onClass :Parent 
] .
```

\[
\geq 2 \text{ hasChild.Parent (John)}
\]

```
:John rdf:type [  
  rdf:type owl:Restriction ;
  owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;
  owl:onProperty :hasChild ;
  owl:onClass :Parent 
] .
```

\[
=3 \text{ hasChild.Parent (John)}
\]

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

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

=5 hasChild. T (John)

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

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

:hasParent owl:inverseOf :hasChild .

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

Orphan $\equiv \forall \text{hasChild}^\neg . \text{Dead}$

: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:FunctionalProperty .

:hasHusband rdf:type owl:InverseFunctionalProperty .

:hasAncestor rdf:type owl:TransitiveProperty .
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 RDF Syntax: Datatypes

```
:personAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:onDatatype xsd:integer;
  owl:withRestrictions (  
    [ xsd:minInclusive "0"^^xsd:integer ]
    [ xsd:maxInclusive "150"^^xsd:integer ]
  )
].

:majorAge 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 )
].
```

Datatype facets
## 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>(axiomatised)</td>
<td>$\top/\bot$</td>
</tr>
<tr>
<td>Class intersection</td>
<td><code>owl:intersectionOf</code></td>
<td>$\land$</td>
<td>$\sqcap$</td>
</tr>
<tr>
<td>Class union</td>
<td><code>owl:unionOf</code></td>
<td>$\lor$</td>
<td>$\sqcup$</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></td>
<td></td>
</tr>
<tr>
<td>Max. cardinality</td>
<td><code>owl:maxQualifiedCardinality</code></td>
<td>$\forall y_1 \ldots y_n+1. \ldots \rightarrow \ldots$</td>
<td>$\leq n \text{ S.C}$</td>
</tr>
<tr>
<td></td>
<td><code>owl:onClass</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local reflexivity</td>
<td><code>owl:hasSelf</code></td>
<td>$R(x,x)$</td>
<td>$\exists R.\text{Self}$</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>⊥</td>
</tr>
<tr>
<td>Inverse</td>
<td>owl:inverseOf</td>
<td>R⁻</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>
<tr>
<td><strong>Property characteristics</strong></td>
<td><strong>rdf:type</strong></td>
<td></td>
</tr>
<tr>
<td>Symmetric</td>
<td>owl:SymmetricProperty</td>
<td>Sym(R)</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>owl:AsymmetricProperty</td>
<td>Asy(R)</td>
</tr>
<tr>
<td>Reflexive</td>
<td>owl:ReflexiveProperty</td>
<td>Ref(R)</td>
</tr>
<tr>
<td>Irreflexive</td>
<td>owl:IrreflexiveProperty</td>
<td>Irr(R)</td>
</tr>
<tr>
<td>Transitivity</td>
<td>owl:TransitiveProperty</td>
<td>Tra(R)</td>
</tr>
</tbody>
</table>

| Subclass              | rdfs:subClassOf                                       | ∀x.C(x) → D(x)   | ⊆∈D   |
| Subproperty           | rdfs:subPropertyOf                                    | ∀x,y.R(x,y) → S(x,y) | ⊆∈S   |
OWL RDF Syntax: Header

```xml
@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/> .
```
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/> .

: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 .
OWL RDF Syntax: Declarations

Each class, property, or individual needs to be declared.

:John  rdf:type owl:NamedIndividual .
:Person  rdf:type owl:Class .
:hasWife  rdf:type owl:ObjectProperty .
:hasAge  rdf:type owl:DatatypeProperty .

Punning:

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

:John  rdf:type  :Father .
:Father  rdf:type  :SocialRole .

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 .

:Man rdfs:subClassOf :Person .
[] rdf:type owl:Axiom ;
   owl:annotatedSource :Man ;
   owl:annotatedProperty rdfs:subClassOf ;
   owl:annotatedTarget :Person ;
   rdfs:comment "States that every man is a person."^^xsd:string .
Contents:

- Overview
- RDF Syntax in Detail
- Other Syntaxes and OWL Variants
**OWL/RDF Semantic Mismatch**

```prolog
ex:speaksWith rdfs:domain ex:Homo .
ex:Homo rdfs:subClassOf ex:Primates .
```

does not RDFS-entail

```prolog
ex:speaksWith rdfs:domain ex:Primates .
```

although it is a valid OWL entailment.

**It does RDFS-entail**

```prolog
rdfs:subClassOf rdf:type rdf:Property
```

which is not a valid OWL entailment.
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 (afaik).
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 Lite$_R$)
- OWL RL (the description logic DLP)
  - inspired by intersecting OWL with Datalog
  - implemented e.g. in Oracle 11g

See course by Markus Krötzsch.
OWL Functional Syntax

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

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

ChildlessPerson ⊆ Person ⊓ ¬∃hasParent⊤.

Person ⊓ ¬Parent (Jack)
OWL Manchester Syntax

Class: Parent
  EquivalentTo: hasChild some Person
  EquivalentTo: Mother or Father

Class: HappyPerson
  EquivalentTo: hasChild only Happy and hasChild some Happy

Class: JohnsChildren
  EquivalentTo: hasParent value John

Class: NarcisticPerson
  EquivalentTo: loves Self

Class: Orphan
  EquivalentTo: inverse hasChild only Dead

Class: Teenager
  SubClassOf: hasAge some integer[<= 13 , >= 19]

Class: X
  SubClassOf: Parent and hasChild max 1 and hasChild only Female
  EquivalentTo: {Mary, Bill, Meg} and Female
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
References

- Patrick Hayes, ed., RDF Semantics. W3C Recommendation, 10 February 2004