Knowledge Representation for the Semantic Web

Winter Quarter 2010

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

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Flyer with special offer is available.

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

<table>
<thead>
<tr>
<th>ex:speaksWith</th>
<th>rdfs:domain</th>
<th>ex:Homo</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex:Homo</td>
<td>rdfs:subClassOf</td>
<td>ex:Primates</td>
</tr>
</tbody>
</table>

**does not RDFS-entail**

| ex:speaksWith | rdfs:domain | ex:Primates |

**although it is a valid OWL entailment.**

It does **RDFS-entail**

| rdfs:subClassOf | rdf:type | rdf:Property |

**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
  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.
Orphan ⊑ Human ⊓ ∀ hasParent. ¬ Alive
• 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 _x5 .`
- `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

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

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

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

{John} ∩ {Bill} ⊆ ⊥
OWL RDF Syntax: Classes + Properties

:Woman rdfs:subClassOf :Person.

:Person owl:equivalentClass :Human.

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

: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 [  
  rdf:type owl:Class ;  
] .

:Grandfather rdfs:subClassOf [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Man :Parent )  
] .
OWL RDF Syntax: Complex Classes

OWL RDF Syntax: Restrictions

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

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

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

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

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

```
JohnsChildren ≡ ∃hasParent.{John}
```

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

```
NarcisticPerson ≡ ∃loves.Self
```

OWL RDF Syntax: Restrictions

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

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

```

≤4 hasChild.Parent (John)

≥2 hasChild.Parent (John)

=3 hasChild.Parent (John)
OWL RDF Syntax: Restrictions

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

=5 hasChild.т (John)


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

:hasParent owl:inverseOf :hasChild .


Orphan $\equiv \forall \text{hasChild} \cdot \text{Dead}$

: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.
Datatype facets

[: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 )
 ]].
# 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>$\top/\bot$</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><strong><code>owl:onProperty</code></strong></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 owl:onClass</code></td>
<td>$\exists y_1 \ldots y_n \ldots$</td>
<td>$\geq n S.C$</td>
</tr>
<tr>
<td>Max. cardinality</td>
<td><code>owl:maxQualifiedCardinality owl:onClass</code></td>
<td>$\forall y_1 \ldots y_n+1. \ldots \to \ldots$</td>
<td>$\leq n S.C$</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><code>owl:propertyChainAxiom</code></td>
<td>—</td>
</tr>
<tr>
<td>Inverse</td>
<td><code>owl:inverseOf</code></td>
<td>$R^{-}$</td>
</tr>
<tr>
<td>Key</td>
<td><code>owl:hasKey</code></td>
<td></td>
</tr>
<tr>
<td>Property disjointness</td>
<td><code>owl:propertyDisjointWith</code></td>
<td>$\text{Dis}(R,S)$</td>
</tr>
<tr>
<td><strong>Property characteristics</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symmetric</td>
<td><code>owl:SymmetricProperty</code></td>
<td>$\text{Sym}(R)$</td>
</tr>
<tr>
<td>Asymmetric</td>
<td><code>owl:AsymmetricProperty</code></td>
<td>$\text{Asy}(R)$</td>
</tr>
<tr>
<td>Reflexive</td>
<td><code>owl:ReflexiveProperty</code></td>
<td>$\text{Ref}(R)$</td>
</tr>
<tr>
<td>Irreflexive</td>
<td><code>owl:IrreflexiveProperty</code></td>
<td>$\text{Irr}(R)$</td>
</tr>
<tr>
<td>Transitivity</td>
<td><code>owl:TransitiveProperty</code></td>
<td>$\text{Tra}(R)$</td>
</tr>
</tbody>
</table>

| Subclass                     | `rdfs:subClassOf`                          | $\forall x.C(x) \rightarrow D(x)$ | $C \sqsubseteq D$ |
| Subproperty                  | `rdfs:subPropertyOf`                       | $\forall x,y.R(x,y) \rightarrow S(x,y)$ | $R \sqsubseteq S$ |

@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#> .

<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 .
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.
owl:RDF Syntax: Annotations

: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 .
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
• 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.
The OWL 2 spec describes three profiles (fragments, sublanguages) which have polynomial complexity.

- OWL EL (the description logic EL++) presented earlier
- OWL QL (the description logic DL Lite$_R$) forthcoming class presentation
- OWL RL (the description logic DLP) skipped
  - inspired by intersecting OWL with Datalog
  - implemented e.g. in Oracle 11g
OWL Functional Syntax

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

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

ChildlessPerson ⊑ Person ⊓ ¬ ∃ hasParent ⊓ T

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
Today’s Session: OWL Syntax

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

• put your corrected DL axioms into your Protege ontology file
• send me the .owl file
• deadline: 3rd of March

• note: this completes two assignments
  – putting ontology into Protege
  – adding the DL axioms

• there will (probably!) be one more project assignment – after the in-class project sessions.
Class project: next week preparation

- Tuesday 3rd of March and Monday 8th of March.

- We will try to do ontology integration.

- Teams of 2-3 people trying to integrate their ontologies – i.e. make one ontology out of them.
  - If we have enough time, we’ll integrate them all!

- You need to do some preparations for this:
  - bring a printout of your ontology (best in Turtle)
  - bring an easy to read version (graphical or other) of the main subclass hierarchy of your ontology.
Today’s Session: OWL Syntax

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. Class Presentations
Class presentations – scheduled

- RDFa – embedding RDF in HTML (W3C standard)
  Pavan, Thursday 28th of January
- Scalable Distributed Reasoning using MapReduce (Urbani, Kotoulas, Oren, van Harmelen, ISWC2009)
  Wenbo, Thursday 28th of January

- TJ, Security and Semantic Web, 1st of March
- Pramod, Virtuoso, 4th of March
- Hemant, FOAF, 4th of March
- Ashutosh, Linked Open Data, 4th of March
- Prateek, SPARQL, 11th of March
- Vinh, Semantic MediaWiki, 11th of March
- Raghava, DL-Lite, 11th of March
03/01/10: OWL part 4 - Tableaux Calculus + 1 class presentation
03/02/10: class project session
03/04/10: 3 class presentations

03/08/10: class project session
03/09/10: exercise session
03/11/10: 3 class presentations

03/12/10: exams