OWL Syntax and Relation to RDF

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

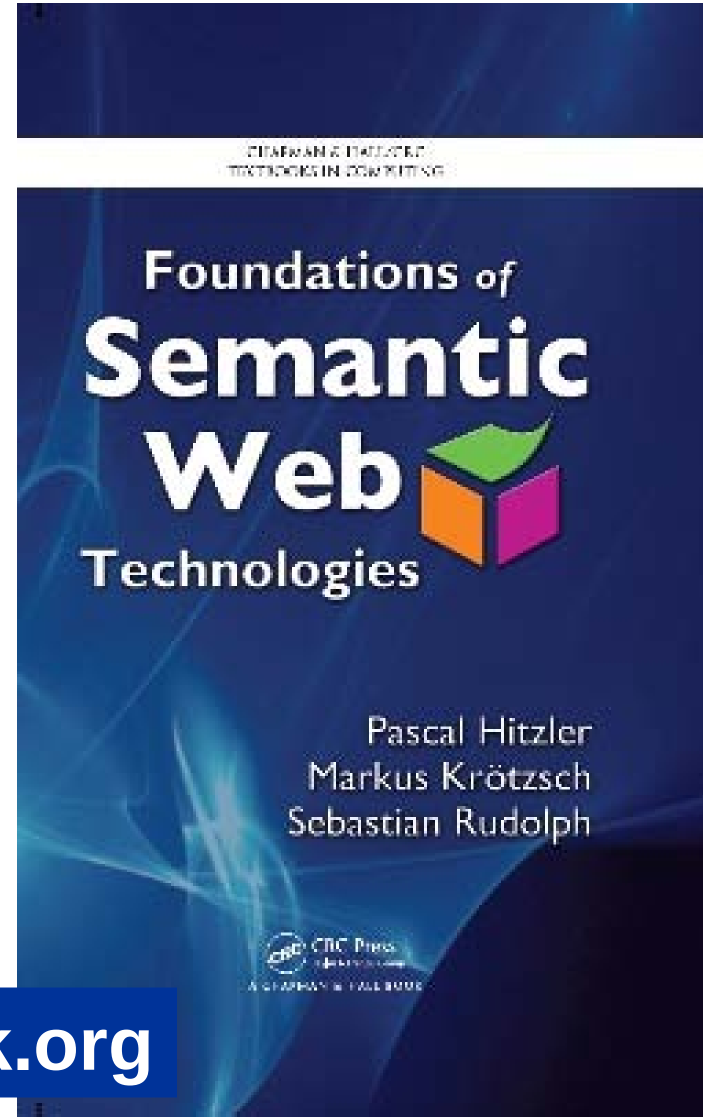
Textbook

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

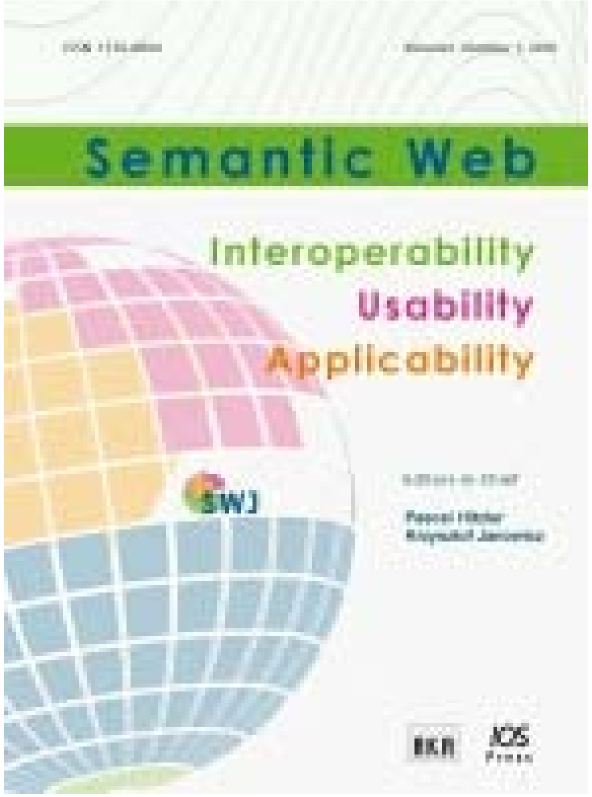
Semantic Web journal

- 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/**

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/**

1. Microsoft
2. Yahoo Research Labs
3. IBM
4. Stanford University
5. Google. Inc.
6. University of Southampton
   Carnegie Mellon University
   VU University of Amsterdam
   Wright State University
7. Chinese Academy of Sciences
   Karlsruhe Institute of Technology
   …

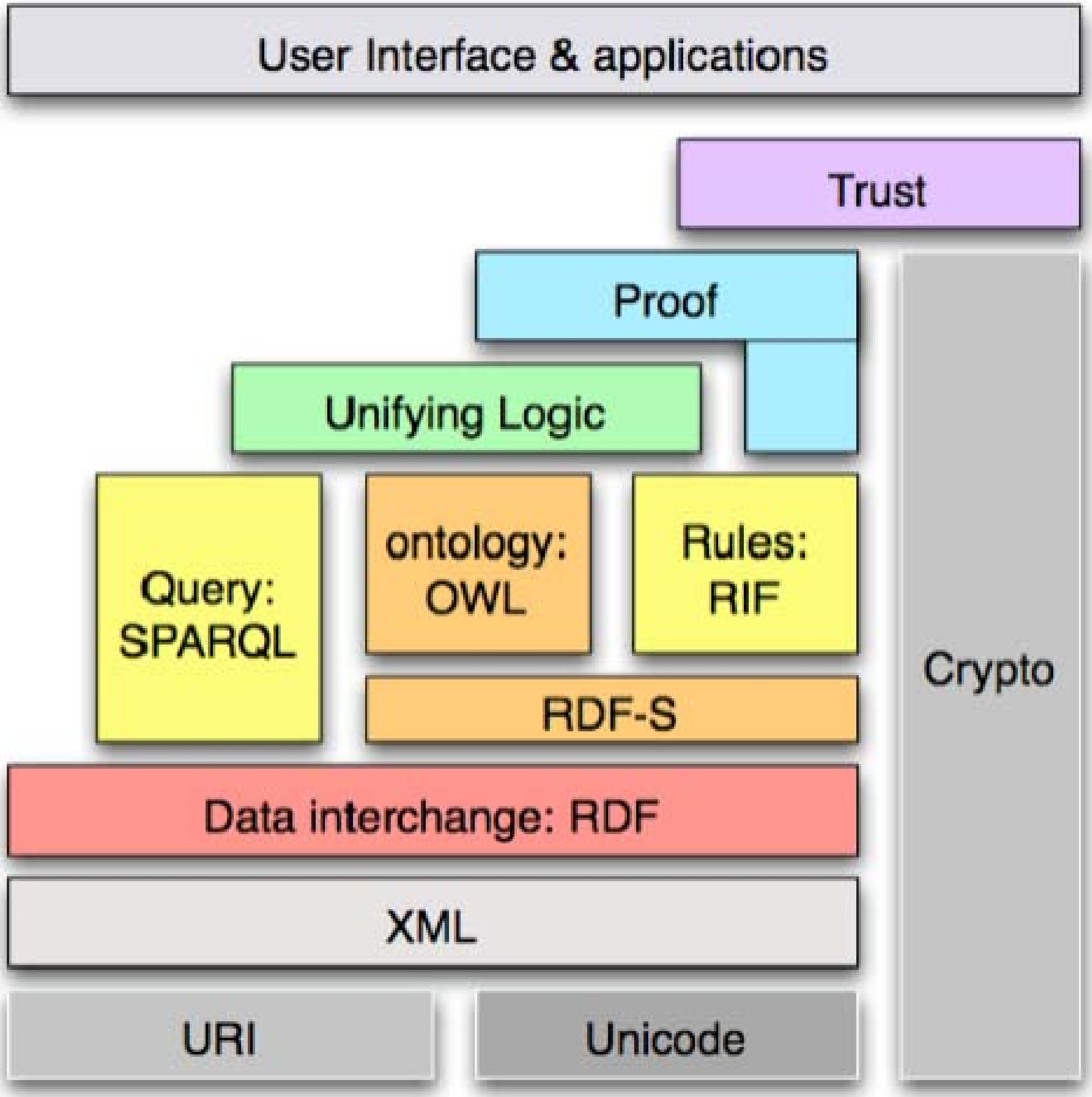
Microsoft Academic Search Beta

In World Wide Web research

out of 2,507 organizations world-wide,

Wright State ranked 6th in January 2013

The Semantic Web Stack



OWL Syntax and Relation to RDF

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
OWL 2 Web Ontology Language: Primer (Second Edition.
W3C Recommendation, 11 December 2012.
http://www.w3.org/TR/owl2-primer/

The RDFS perspective

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

- Person(mary)
- Mother $\sqsubseteq$ Woman
- hasWife(john,mary)
- hasWife $\sqsubseteq$ hasSpouse

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

- $\top \sqsubseteq \forall$hasWife.Woman
- $\top \sqsubseteq \forall$hasWife^{-}.Man or $\exists$hasWife.$\top \sqsubseteq$ Man

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

- $\top$
- $\bot$
- U
- *?*

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
  $\sqcap$, $\sqcup$, $\neg$, $\exists$, $\forall$
  $\top$, $\bot$
- 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**

$$\mathtt{Orphan} \sqsubseteq \mathtt{Human} \sqcap \forall \mathtt{hasParent}.\neg \mathtt{Alive}$$

  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.**

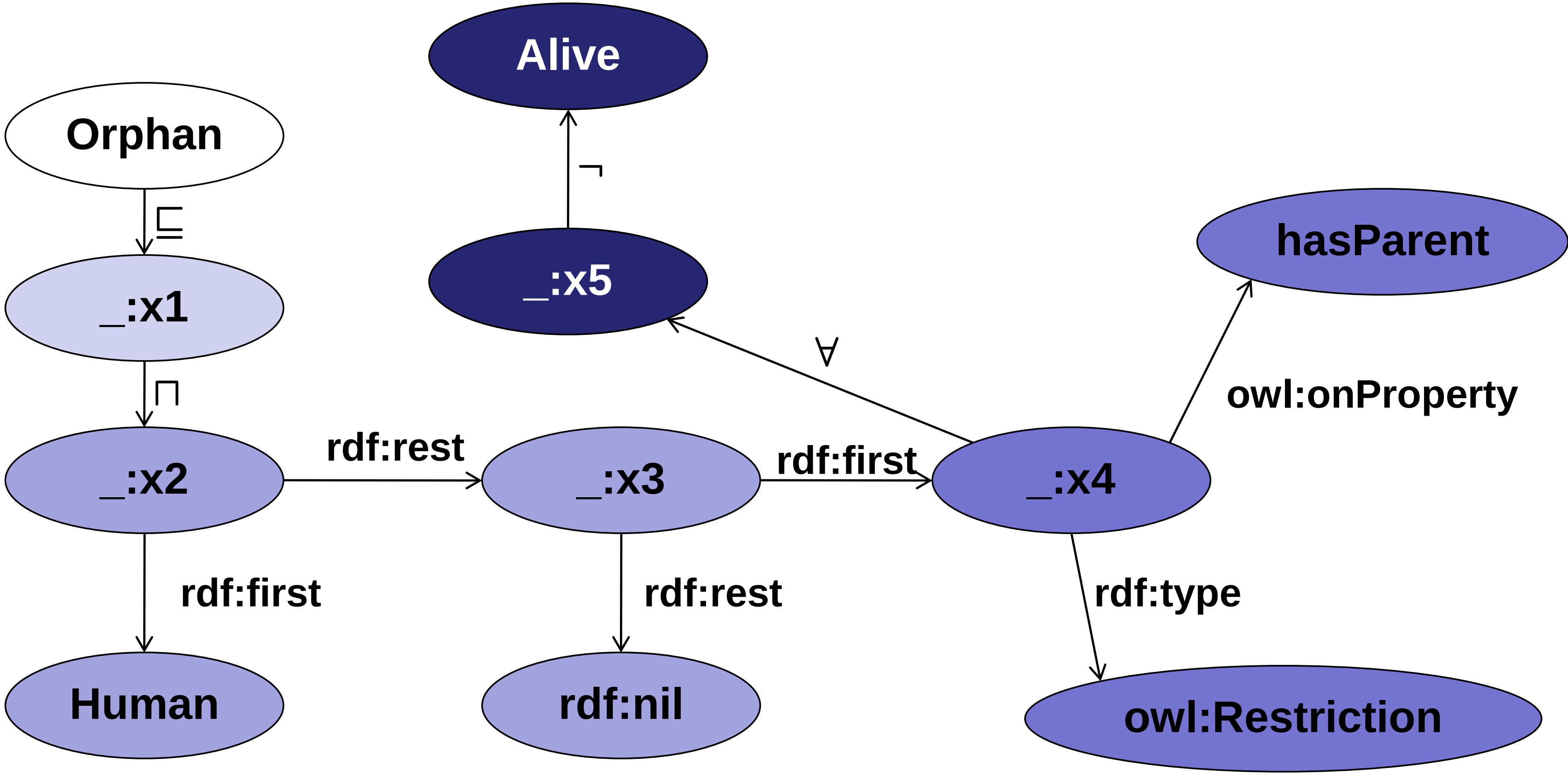
OWL Syntax and Relation to RDF

Contents:

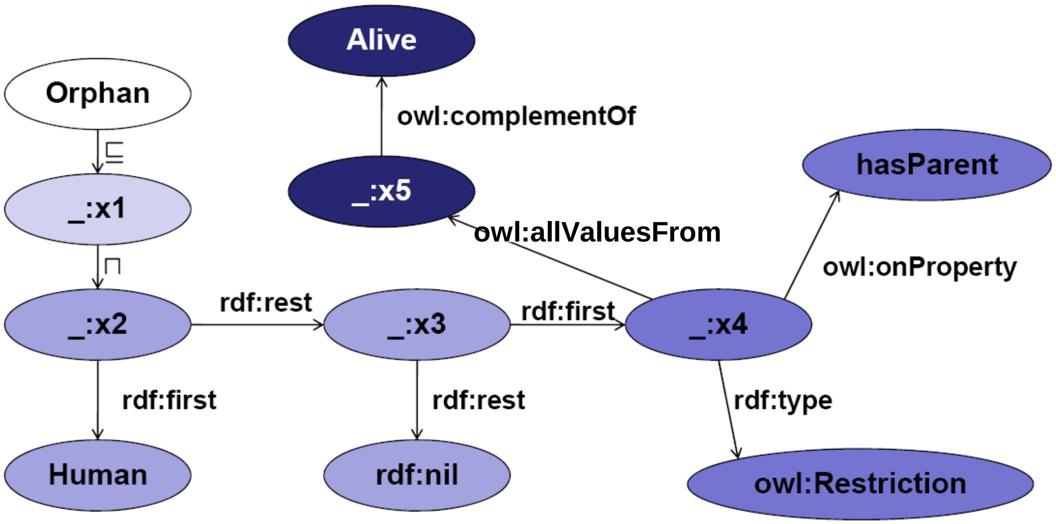
- **Overview**
- **RDF Syntax in Detail**
- **Other Syntaxes and OWL Variants**

General Idea

$\texttt{Orphan} \sqsubseteq \texttt{Human} \sqcap \forall\texttt{hasParent}.\neg\texttt{Alive}$



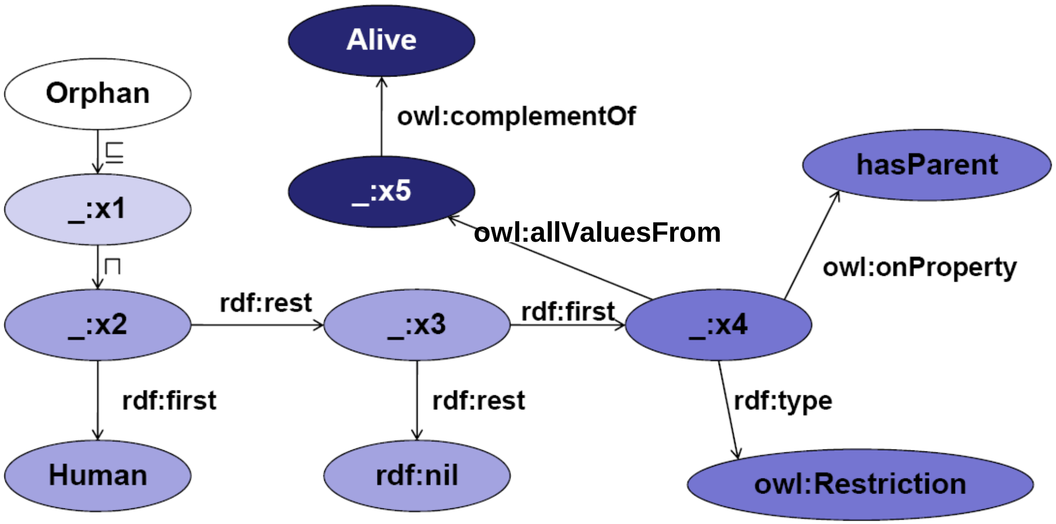
RDF Syntax



- 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 .
```

$\{John\} \sqcap \{Bill\} \sqsubseteq \bot$

```
:James owl:sameAs :Jim.
```

$\{John\} \equiv \{Jim\}$

```
:John  :hasAge ”51”^^xsd:nonNegativeInteger .
```

```
[]  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 ) .
```

Woman $\sqcap$ 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 )
] .
```

Mother ≡ Woman ⊓ Parent

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

Parent ≡ Mother ⊔ Father

```
:ChildlessPerson  owl:equivalentClass  [
  rdf:type              owl:Class ;
  owl:intersectionOf  ( :Person  [ owl:complementOf  :Parent ] )
] .
```

ChildlessPerson ≡ Person ⊓ ¬Parent

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

OWL RDF Syntax: Complex Classes

Person ⊓ ¬Parent (Jack)

```
:Jack  rdf:type  [
  rdf:type            owl:Class ;
  owl:intersectionOf  ( :Person
                        [ rdf:type          owl:Class ;
                          owl:complementOf  :Parent     ]
                      )
] .
```

OWL RDF Syntax: Restrictions

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

Parent $\equiv$ $\exists$hasChild.Person

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

Orphan $\equiv \forall$ hasChild^{-}.Dead

OWL RDF Syntax: Restrictions

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

JohnsChildren $\equiv$ $\exists$hasParent.{John}

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

NarcisticPerson $\equiv$ $\exists$loves.Self

OWL RDF Syntax: Restrictions

≤4 hasChild.Parent (John)

```
: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
] .
```

≥2 hasChild.Parent (John)

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

=3 hasChild.Parent (John)

OWL RDF Syntax: Restrictions

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

$=5\ \text{hasChild}.\top\ (\text{John})$

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

$\text{MyBirthdayGuests} \equiv \{\text{Bill}, \text{John}, \text{Mary}\}$

OWL RDF Syntax: Properties

```
:hasParent owl:inverseOf :hasChild .
```

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

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

```
: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 .
```

OWL RDF Syntax: Properties

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

$$\mathbf{hasParent} \circ \mathbf{hasParent} \sqsubseteq \mathbf{hasGrandParent}$$

```
:Person owl:hasKey ( :hasSSN ) .
```

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 ]
   )
 ] .
```

Datatype *facets*

```
: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

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	`owl:Thing`/`owl:Nothing`	(axiomatise)	$\top/\bot$
Class intersection	`owl:intersectionOf`	$\wedge$	$\sqcap$
Class union	`owl:unionOf`	$\vee$	$\sqcup$
Class complement	`owl:complementOf`	$\neg$	$\neg$
Enumerated class	`owl:oneOf`	(ax. with $\approx$)	$\{a\}$
Property restrictions	`owl:onProperty`		
Existential	`owl:someValueFrom`	$\exists y \ldots$	$\exists R.C$
Universal	`owl:allValuesFrom`	$\forall y \ldots$	$\forall R.C$
Min. cardinality	`owl:minQualifiedCardinality` `owl:onClass`	$\exists y1\ldots yn. \ldots$	$\geq n\ S.C$
Max. cardinality	`owl:maxQualifiedCardinality` `owl:onClass`	$\forall y1\ldots yn{+}1. \ldots \rightarrow \ldots$	$\leq n\ S.C$
Local reflexivity	`owl:hasSelf`	$R(x,x)$	$\exists R.Self$

Essential OWL Features

Feature	Related OWL vocabulary	DL
Property chain	`owl:propertyChainAxiom`	$\circ$
Inverse	`owl:inverseOf`	R^-
Key	`owl:hasKey`	
Property disjointness	`owl:propertyDisjointWith`	Dis(R,S)
Property characteristics	rdf:type	
Symmetric	`owl:SymmetricProperty`	Sym(R)
Asymmetric	`owl:AsymmetricProperty`	Asy(R)
Reflexive	`owl:ReflexiveProperty`	Ref(R)
Irreflexive	`owl:IrreflexiveProperty`	Irr(R)
Transitivity	`owl:TransitiveProperty`	Tra(R)

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

OWL RDF Syntax: Header

```
<http://example.com/owl/families> rdf:type owl:Ontology .
```

```
@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.

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 .
```

OWL Syntax and Relation to RDF

Contents:

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

OWL/RDF Semantic Mismatch

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

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 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).

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 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
      )
    )
  )
)
```

$\text{ChildlessPerson} \sqsubseteq \text{Person} \sqcap \neg \exists \text{hasParent}^{-}.\top$

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

$\text{Person} \sqcap \neg\text{Parent} (\text{Jack})$

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
```

OWL Manchester Syntax

```
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

- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies. Textbooks in Computing, Chapman and Hall/CRC Press, 2009.
- Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation, 11 December 2012.
- Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds, OWL 2 Web Ontology Language: Direct Semantics (Second Edition). W3C Recommendation, 11 December 2012
- Patrick Hayes, ed., RDF Semantics. W3C Recommendation, 10 February 2004