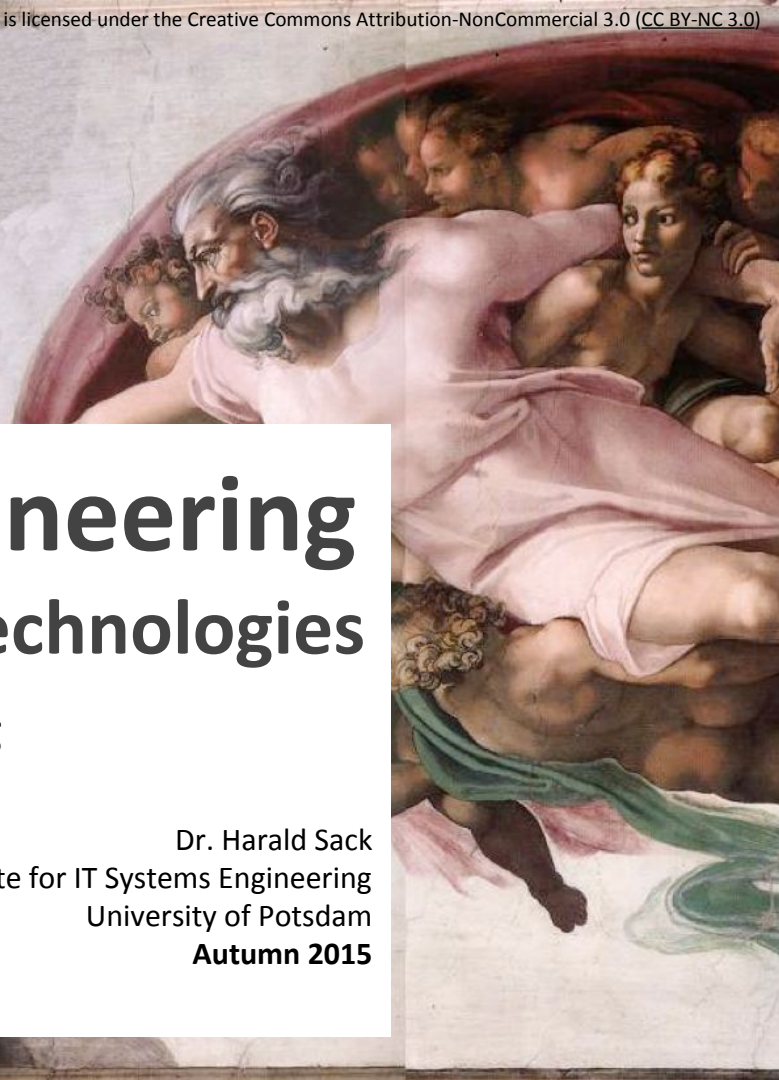
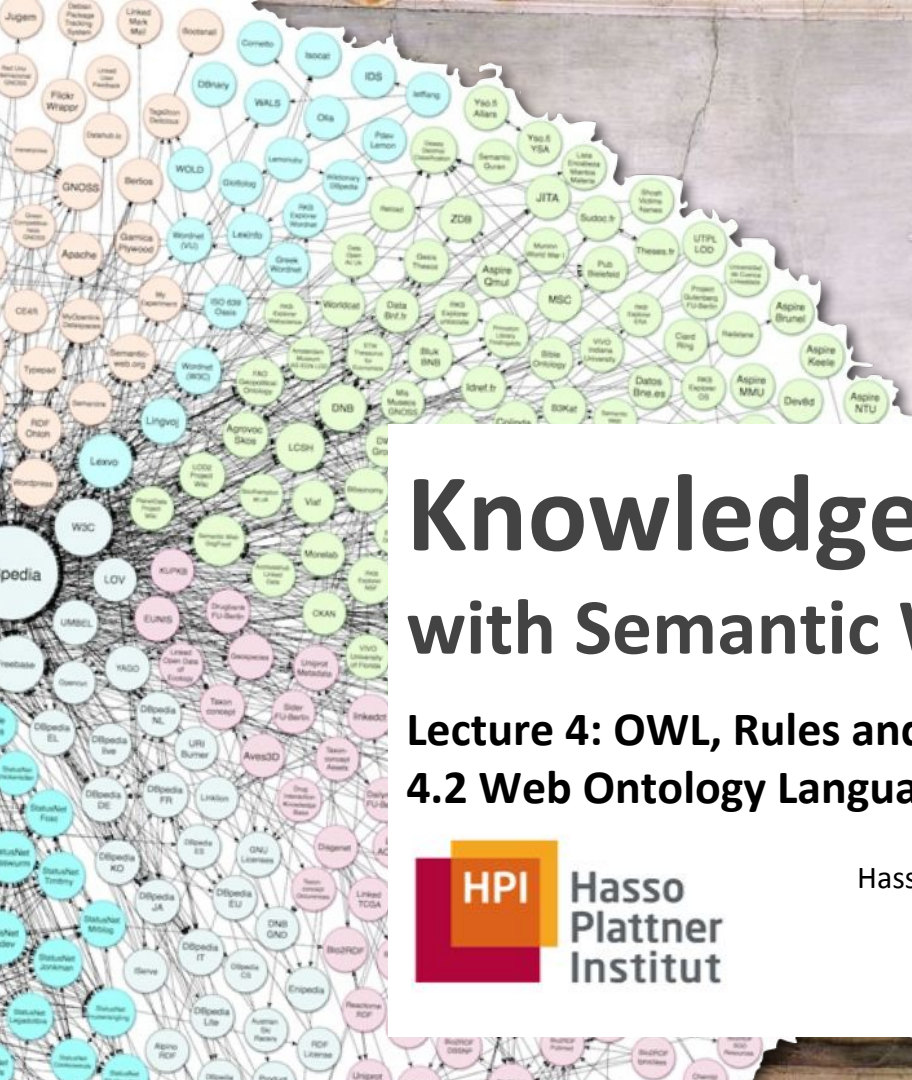




Knowledge Engineering with Semantic Web Technologies

Lecture 4: OWL, Rules and Reasoning 4.2 Web Ontology Language - OWL

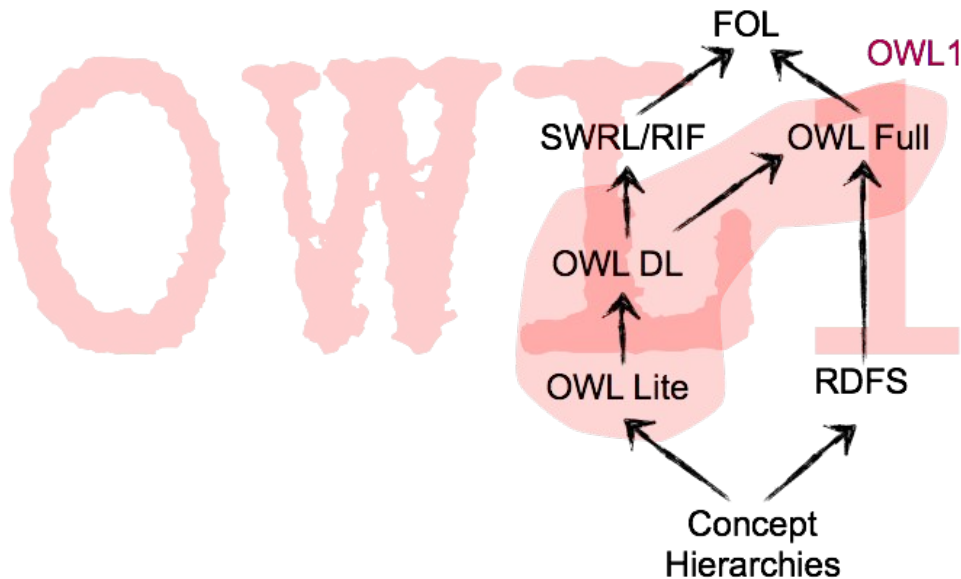


Dr. Harald Sack
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Autumn 2015

- OWL [*SHOIN*($\mathcal{D}$)] W3C Recommendation since 2004
- OWL 2 [*SROIQ*($\mathcal{D}$)] W3C Recommendation since 2009
- an OWL Ontology consists of
 - classes / properties / individuals (instances of classes)
- Open World Assumption
 - „Absence of information must not be valued as negative information.“
 - E.g.: `sitsNextTo(PersonA, PersonB)`
PersonA may also sit next to another person...
- No Unique Name Assumption
 - „Difference must be expressed explicitly“
E.g.: PersonA possibly denotes the same individual as PersonB

Web Ontology Language - OWL

- OWL is a semantic fragment of FOL
- OWL1 exists in different flavors
 - $\text{OWL Lite} \subseteq \text{OWL1 DL} \subseteq \text{OWL1 Full}$



OWL1 is based on *SHOIN(D)*

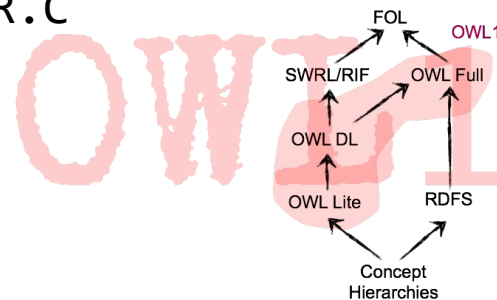
- **Axioms**

- **TBox**: subclass relationships $C \sqsubseteq D$ ($\mathcal{H}$)
- **RBox**: subproperty relationships $R \sqsubseteq S$ ($\mathcal{H}$),
inverse properties R^- ($\mathcal{I}$), transitivity $\sqsubseteq^+$ ($\mathcal{S}$)
- **ABox**: facts for classes $C(a)$, properties $R(a, b)$,
equality $a=b$, difference $a \neq b$

- **Class constructors:**

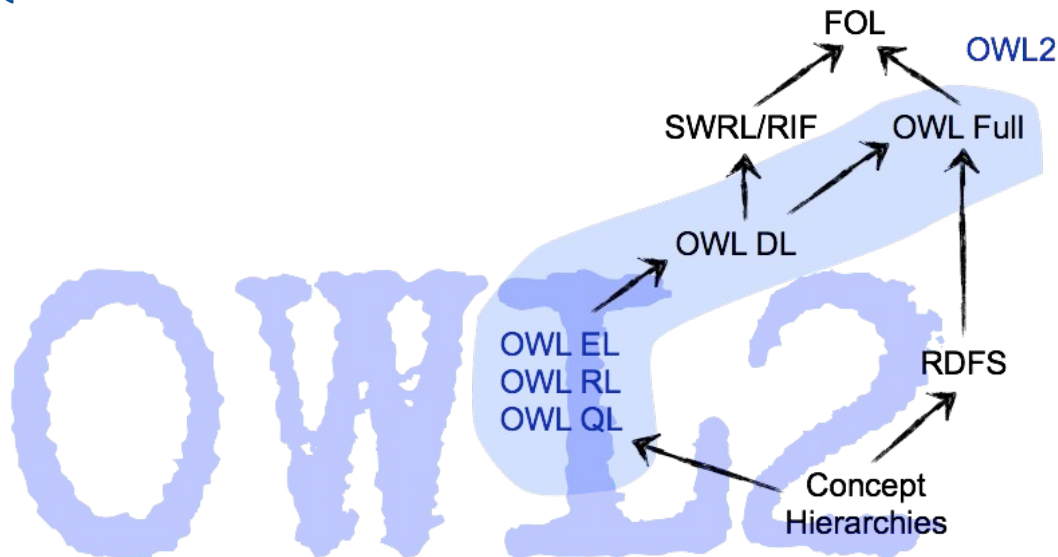
- conjunction $C \sqcap D$, disjunction $C \sqcup D$, negation $\neg C$ of classes
- property restrictions: universal $\forall R.C$ and existential $\exists R.C$
- number restrictions: $\leq n R$ and $\geq n R$ ($\mathcal{N}$)
- closed classes (nominals): $\{a\}$ ($\mathcal{O}$)

- **Datatypes** ($\mathcal{D}$)



Web Ontology Language - OWL

- OWL is a semantic fragment of FOL
- OWL2 also exists in different flavors
 - $\text{OWL EL, OWL RL, OWL QL} \subseteq \text{OWL2 DL} \subseteq \text{OWL2 Full}$



OWL2 is based on $\mathcal{SROIQ}(\mathcal{D})$

Class Expressions

- Class names A, B
- Conjunction $C \sqcap D$
- Disjunction $C \sqcup D$
- Negation $\neg C$
- Exist. property restriction $\exists R.C$
- Univ. property restriction $\forall R.C$
- Self $\exists S.\text{Self}$
- Greater-than $\geq n \ S.C$
- Less-than $\leq n \ S.C$
- Enumerated classes $\{a\}$

Properties

- Property names R, S, T
- Simple properties S, T
- Inverse properties R^{-}
- Universal property U

Tbox (Class axioms)

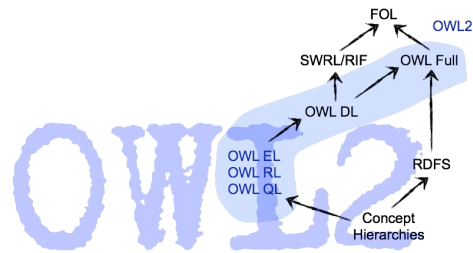
- Inclusion $C \sqsubseteq D$
- Equivalence $C \equiv D$

Rbox (Property Axioms)

- Inclusion $R_1 \sqsubseteq R_2$
- General Inclusion $R^{(-)}_1 \circ R^{(-)}_2 \circ \dots \circ R^{(-)}_n \sqsubseteq R$
- Transitivity
- Symmetry
- Reflexivity
- Irreflexivity
- Disjointness

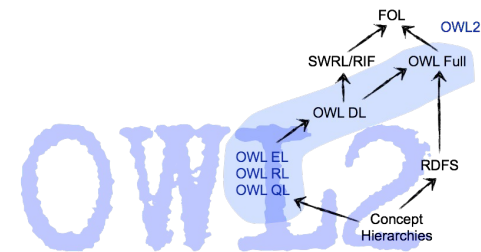
Abox (Facts)

- Class membership $C(a)$
- Property relation $R(a, b)$
- Negated property relation $\neg S(a, b)$
- Equality $a=b$
- Inequality $a \neq b$



OWL Syntax Variants

- OWL2 can be represented in different Syntax variants
 - **Functional Syntax:** substitutes abstract syntax of OWL1
 - **RDF/XML-Syntax:** extension of existing OWL/RDF
 - **OWL/XML-Syntax:** independent XML serialisation
 - **Manchester-Syntax:** machine readable syntax, esp. for ontology editors
 - **Turtle:** concise and easy readable

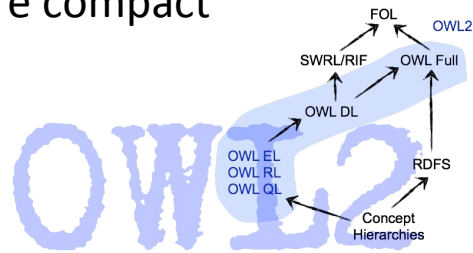


OWL2 Functional Syntax Example

$\text{HappyPerson} \equiv \forall \text{hasChild}.\text{HappyPerson} \sqcap \exists \text{hasChild}.\text{HappyPerson}$

```
Ontology(  
  Declaration(Class(:HappyPerson))  
  EquivalentClasses(:HappyPerson  
    ObjectIntersectionOf(  
      ObjectAllValuesFrom(:hasChild :HappyPerson)  
      ObjectSomeValuesFrom(:hasChild :HappyPerson)  
    )  
  )  
)
```

- Functional Syntax is easy to define, no RDF restrictions, more compact

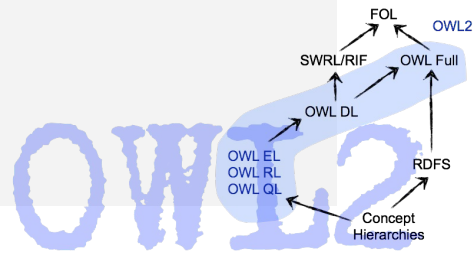


OWL2 RDF/XML Syntax Example

HappyPerson $\equiv \forall \text{hasChild}.\text{HappyPerson} \sqcap \exists \text{hasChild}.\text{HappyPerson}$

```
<?xml version="1.0"?>
<rdf:RDF xmlns=...>
  <owl:ObjectProperty rdf:about="http://example.org/turtle#hasChild"/>

  <owl:Class rdf:about="http://example.org/turtle#HappyPerson">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="http://example.org/turtle#hasChild"/>
            <owl:allValuesFrom rdf:resource="http://example.org/turtle#HappyPerson"/>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="http://example.org/turtle#hasChild"/>
            <owl:someValuesFrom rdf:resource="http://example.org/turtle#HappyPerson"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
</rdf:RDF>
```

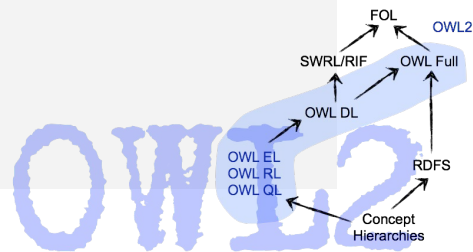


OWL2 OWL/XML Syntax Example

HappyPerson $\equiv \forall \text{hasChild}.\text{HappyPerson} \sqcap \exists \text{hasChild}.\text{HappyPerson}$

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"....>

  <Declaration>
    <Class abbreviatedIRI=":HappyPerson"/>
  </Declaration>
  <EquivalentClasses>
    <Class abbreviatedIRI=":HappyPerson"/>
      <ObjectIntersectionOf>
        <ObjectAllValuesFrom>
          <ObjectProperty abbreviatedIRI=":hasChild"/>
          <Class abbreviatedIRI=":HappyPerson"/>
        </ObjectAllValuesFrom>
        <ObjectSomeValuesFrom>
          <ObjectProperty abbreviatedIRI=":hasChild"/>
          <Class abbreviatedIRI=":HappyPerson"/>
        </ObjectSomeValuesFrom>
      </ObjectIntersectionOf>
    </EquivalentClasses>
  </Ontology>
```



OWL2 Manchester Syntax Example

`HappyPerson  $\equiv \forall \text{hasChild}.\text{HappyPerson} \sqcap \exists \text{hasChild}.\text{HappyPerson}$`

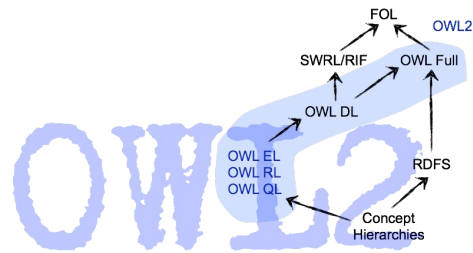
Ontology:

ObjectProperty: `hasChild`

Class: `HappyPerson`

EquivalentTo:

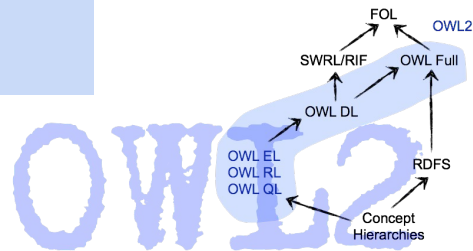
`(hasChild only HappyPerson)`
`and (hasChild some HappyPerson)`



OWL2 Turtle Syntax Example

HappyPerson $\equiv \forall \text{hasChild}.\text{HappyPerson} \sqcap \exists \text{hasChild}.\text{HappyPerson}$

```
:HappyPerson a owl:Class ;  
  owl:equivalentClass [  
    a owl:Class ;  
    owl:intersectionOf ([ a owl:Restriction ;  
      owl:onProperty :hasChild ;  
      owl:allValuesFrom :HappyPerson ]  
      [ a owl:Restriction ;  
        owl:onProperty :hasChild ;  
        owl:someValuesFrom :HappyPerson ]  
    )  
  ] .
```





03 OWL Classes, Properties, and Individuals

OpenHPI - Course Knowledge Engineering with Semantic Web Technologies

Lecture 4: OWL, Rules, and Reasoning