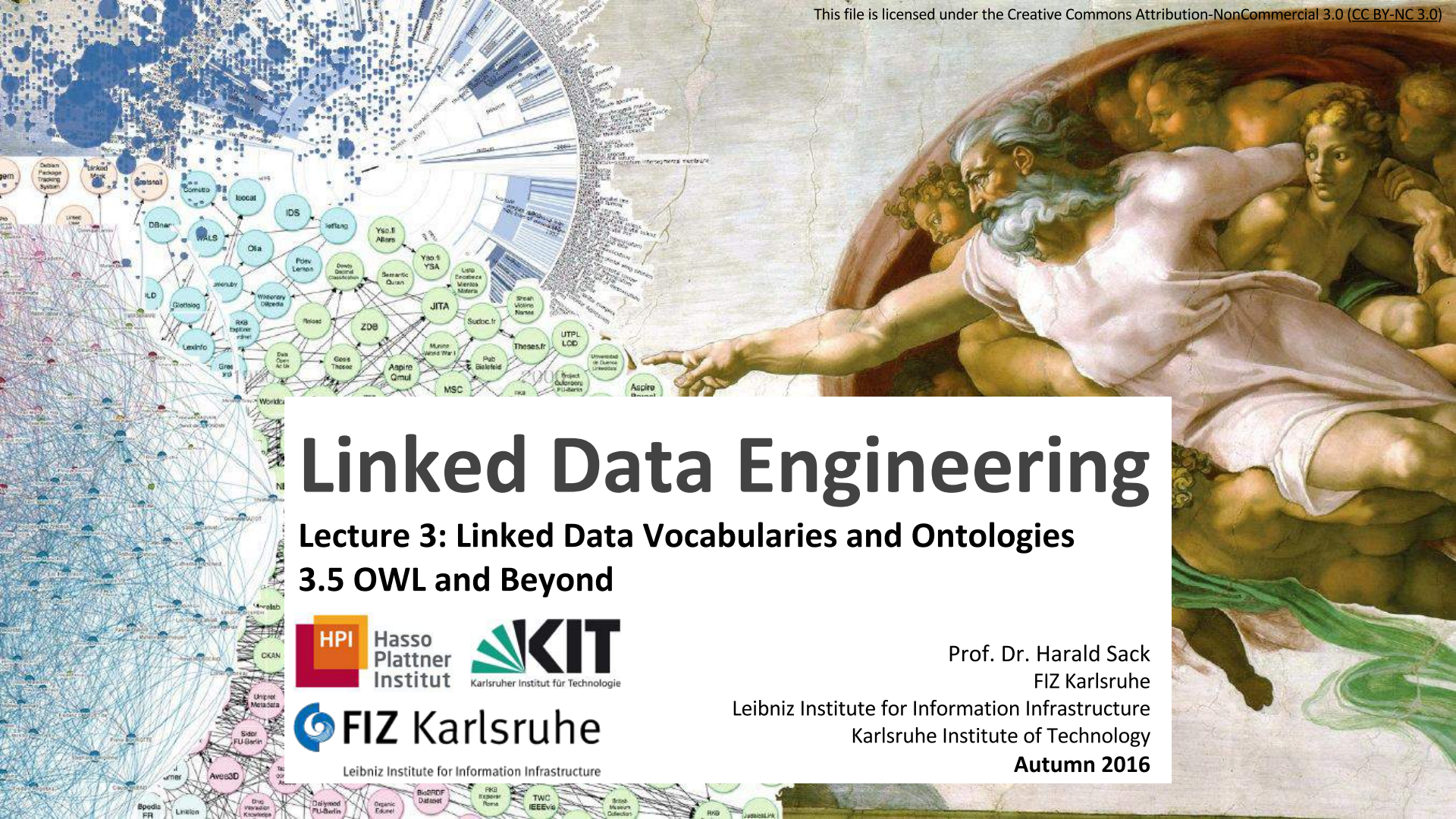




Linked Data Engineering

Lecture 3: Linked Data Vocabularies and Ontologies 3.5 OWL and Beyond



Leibniz Institute for Information Infrastructure

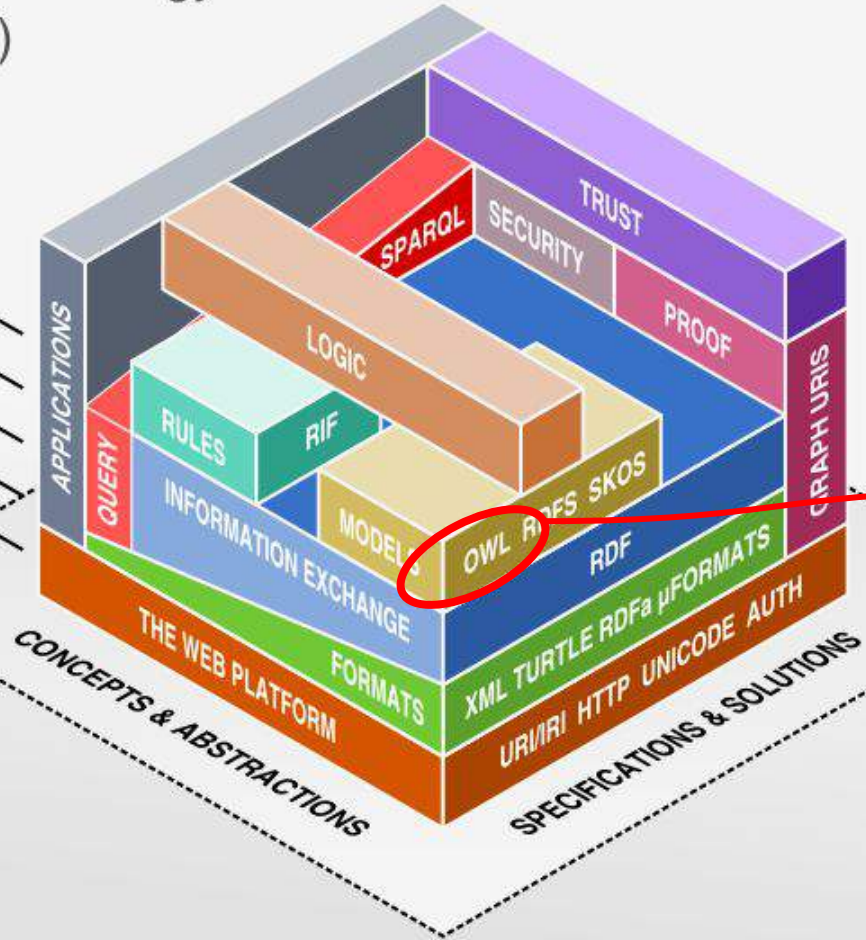
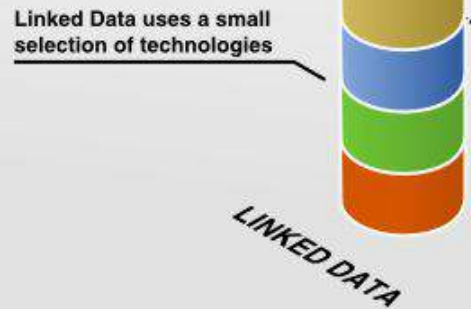
Prof. Dr. Harald Sack
FIZ Karlsruhe

Leibniz Institute for Information Infrastructure
Karlsruhe Institute of Technology

Autumn 2016

The Semantic Web Technology Stack (not a piece of cake...)

- Most apps use only a subset of the stack
- Querying allows fine-grained data access
- Standardized information exchange is key
- Formats are necessary, but not too important
- The Semantic Web is based on the Web



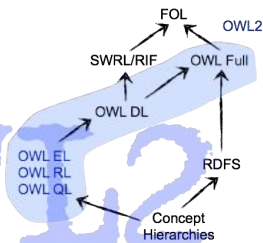
Web Ontology Language (OWL)

OWL Property Relationships

- **Property hierarchies** can be created via specializations: `rdfs:subPropertyOf`
- **Inverse properties** are defined via `owl:inverseOf`
- **Identical properties** are defined via `owl:equivalentProperty`

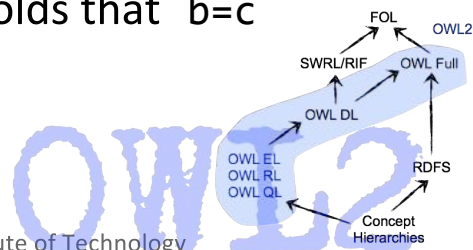
```
:isMadeOf a owl:ObjectProperty ;  
    rdfs:subPropertyOf :consistsOf .  
  
:reads a owl:ObjectProperty ;  
    owl:inverseOf :isReadBy .  
  
:composedOf a owl:ObjectProperty ;  
    owl:equivalentProperty :consistsOf .
```

isMadeOf $\sqsubseteq$ consistsOf
reads- $\equiv$ isReadBy
composedOf $\equiv$ consistsOf



OWL Property Relationships

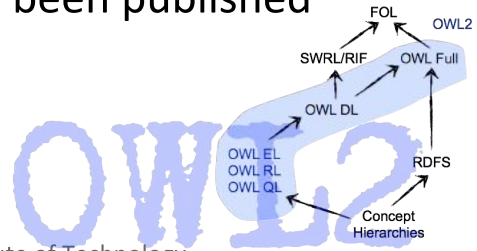
- `owl:TransitiveProperty`
e.g.: if `isPartOf(a,b)` and `isPartOf(b,c)` then it holds that `isPartOf(a,c)`
- `owl:SymmetricProperty`
e.g.: if `isNeighborOf(a,b)` then it holds that `isNeighborOf(b,a)`
- `owl:FunctionalProperty`
e.g.: if `hasMother(a,b)` and `hasMother(a,c)` then it holds that `b=c`
- `owl:InverseFunctionalProperty`
e.g.: if `isMotherOf(b,a)` and `isMotherOf(c,a)` then it holds that `b=c`



OWL Transitive Properties

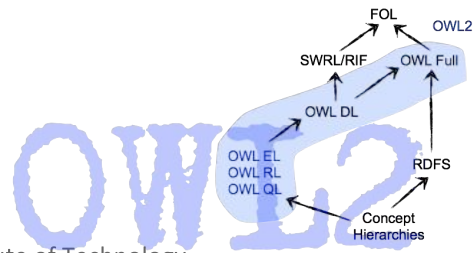
```
:isPublishedBefore a owl:ObjectProperty ;  
                    a owl:TransitiveProperty ;  
                    rdfs:domain owl:Book ;  
                    rdfs:range owl:Book .  
  
:AnimalFarm a owl:Book ;  
             :isPublishedBefore :NineteenEightyfour .  
  
:BraveNewWorld a :Book ;  
               :isPublishedBefore :AnimalFarm .
```

- via inference it can be entailed that `:BraveNewWorld` has been published before `:NineteenEightyfour`



OWL More Property Relationships

- **Asymmetric properties** via `owl:AsymmetricProperty`
e.g.: if it holds that `isLeftOf(a,b)`
then it is not possible that `isLeftOf(b,a)`
- **Reflexive properties** via `owl:ReflexiveProperty`
e.g.: `isRelatedTo(x,x)`
- **Irreflexive properties** via `owl:IrreflexiveProperty`
e.g.: if `isParentOf(x,y)` then $x \neq y$



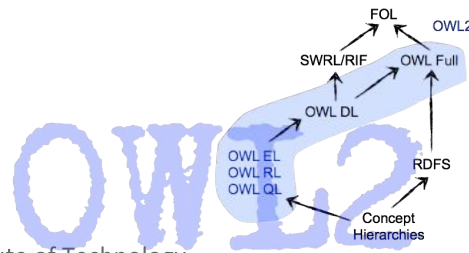
OWL Disjunctive Properties

- Two properties R and S are **disjunctive**,
if two individuals x,y are never related via both properties

```
:hasParent a owl:ObjectProperty ;  
    owl:propertyDisjointWith :hasChild .
```

- Shortcut** for several **disjunctive properties**

```
[] rdf:type owl:AllDisjointProperties ;  
    owl:members  
        ( :hasParent  
          :hasChild  
          :hasGrandchild ) .
```

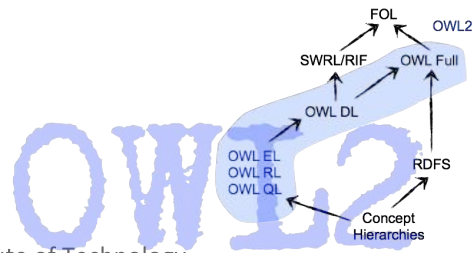


OWL Negated Property Instantiation

- Two individuals can explicitly defined to be **not related with each other** via a given property

`-isBrother(GeorgeOrwell,AldousHuxley)`

```
[ ] rdf:type owl:NegativePropertyAssertion ;  
    owl:sourceIndividual :GeorgeOrwell ;  
    owl:assertionProperty :isBrother ;  
    owl:targetIndividual :AldousHuxley .
```



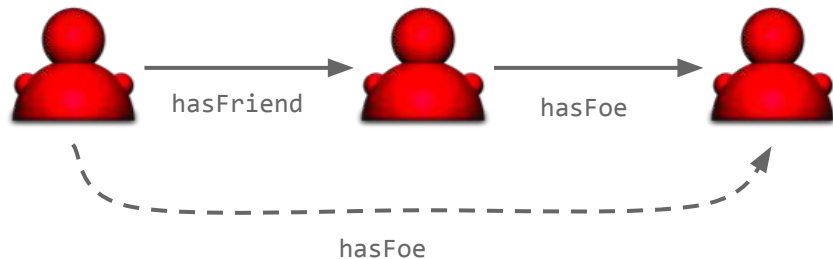
- **Complex Roles** (properties) can be constructed from simple roles (**RBox**)

- „*The friends of my friends are also my friends.*“

`hasFriend a owl:TransitiveProperty .`

- But: „*The foes of my friends are also my foes.*“

- cannot be expressed in that way



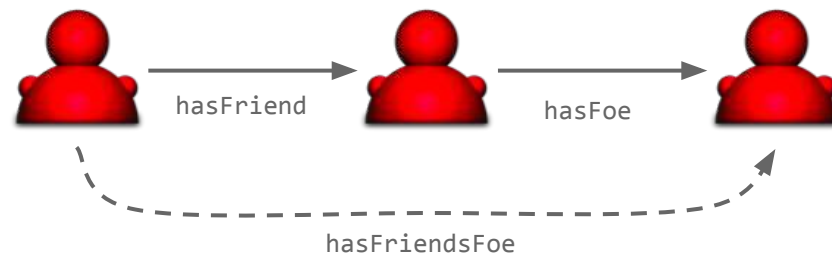
- in FOL it can be expressed as a rule (axiom):

- $\forall x, y, z: \text{hasFriend}(x, y) \wedge \text{hasFoe}(y, z) \rightarrow \text{hasFriendsFoe}(x, z)$

- **General Role Inclusion (property chaining)**

```
:hasFriendsFoe a owl:ObjectProperty ;  
  owl:propertyChainAxiom ( :hasFriend :hasFoe ) .
```

- not allowed for datatype properties



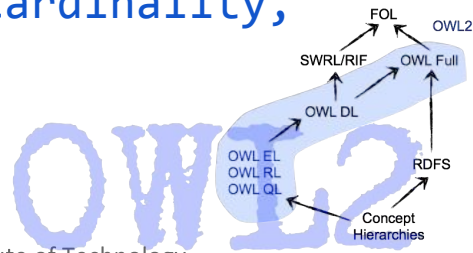
OWL Qualified Number Restrictions

- Class constructors with **number restrictions on properties connected with a range constraint**

SuccessfulAuthor $\sqsubseteq \geq 1$ notableWork.Bestseller

```
:SuccessfulAuthor a owl:Class;  
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:onProperty :notableWork ;  
    owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ;  
    owl:onClass :Bestseller ] .
```

- `owl:maxQualifiedCardinality`, `owl:minQualifiedCardinality`,
`owl:qualifiedCardinality`

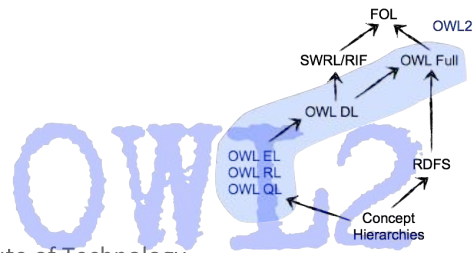


OWL Reflexive Property Restriction

- Classes that contain individuals that are **related to themselves** for specific properties

Philosopher $\sqsubseteq$ knows.self

```
:Philosopher a owl:Class ;  
  rdfs:subClassOf [  
    a owl:Restriction ;  
    owl:onProperty :knows ;  
    owl:hasSelf "true"^^xsd:boolean ] .
```



- More **expressivity** also means more **complexity**
 - this might lead to **undecidability** (as for FOL)
- Do we really need more expressivity than OWL DL offers?
- Consider the following example:
 - „*A squanderer is a person whose expenses are higher than his income*“

- Squanderer $\sqsubseteq$ Person
- Squanderer $\sqsubseteq$ hasExpenses. $\top$
- Squanderer $\sqsubseteq$ hasIncome. $\top$

?

- We need a constructor to combine Classes and Properties
- **Problem:** Mixing of TBox and ABox

- The following example can be expressed via a **FOL-Rule**:
 - „*A squanderer is a person whose expenses are higher than his income*“

```
Person(x) ∧ hasIncome(x,y)
           ∧ hasExpenses(x,z)
           ∧ (z > y)
           → Squanderer(x)
```

- Arithmetics can be part of rules and modeled like a predicate
 - $(z > y) \hat{=}$ greaterThan(z,y)



Next: 06 - RDF(S) and OWL Knowledge Bases

Lecture 3: Linked Data Vocabularies and Ontologies - OpenHPI - Course Linked Data Engineering