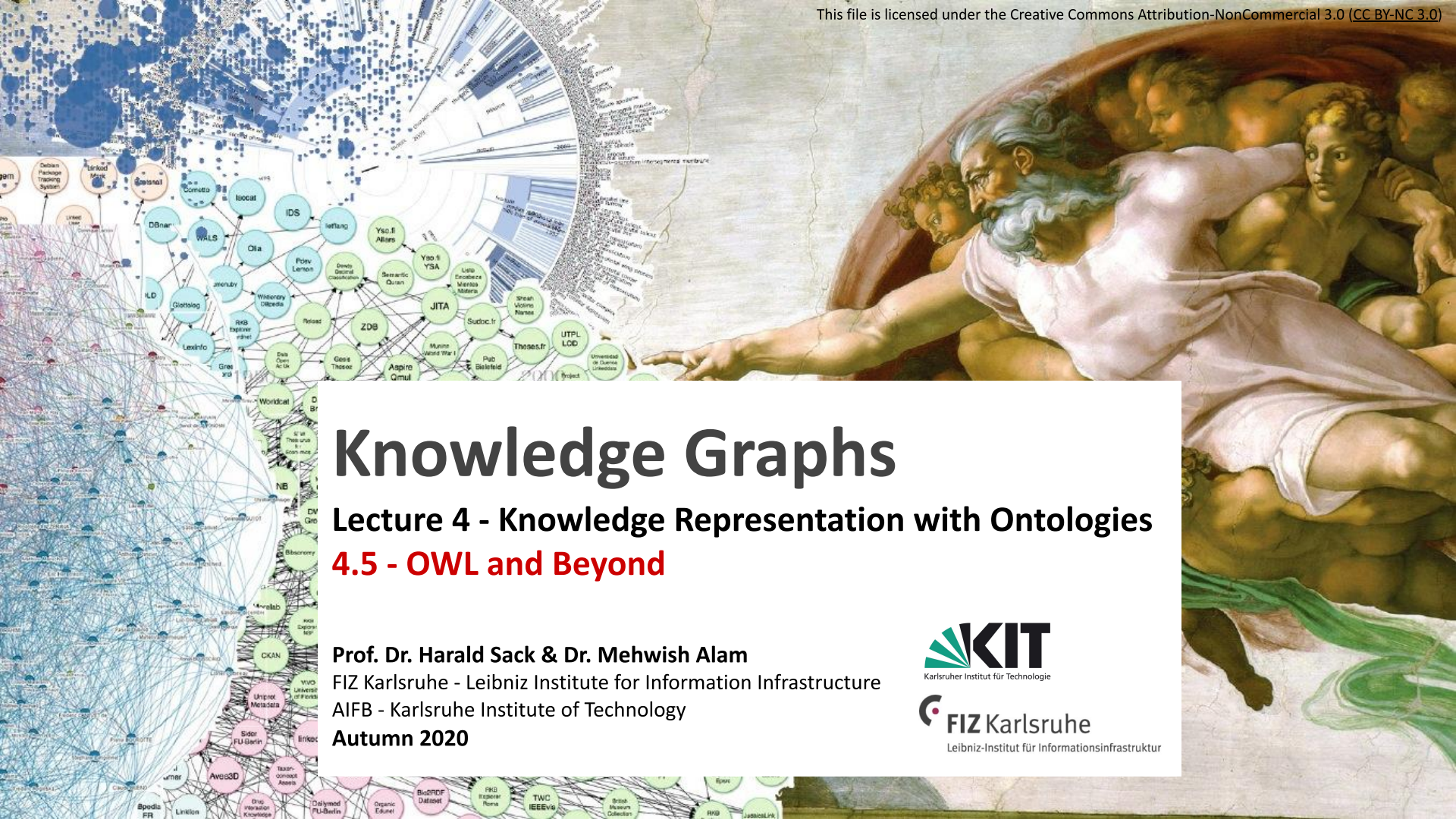


Knowledge Graphs

Lecture 4 - Knowledge Representation with Ontologies

4.5 - OWL and Beyond

Prof. Dr. Harald Sack & Dr. Mehwish Alam
FIZ Karlsruhe - Leibniz Institute for Information Infrastructure
AIFB - Karlsruhe Institute of Technology
Autumn 2020

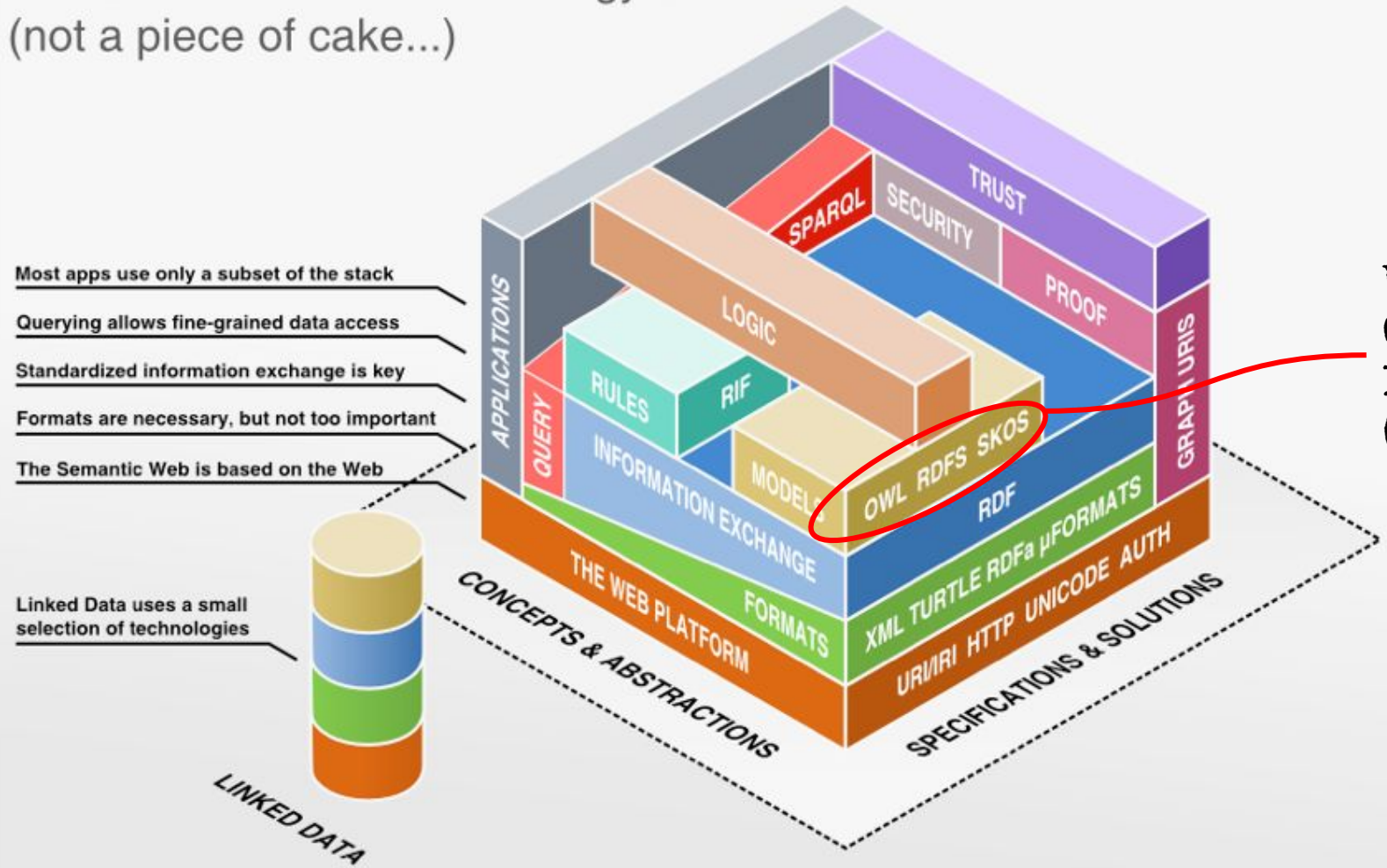
Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz-Institut für Informationsinfrastruktur

Knowledge Graphs

Lecture 4: Knowledge Representation with Ontologies

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



Web Ontology Language (OWL)

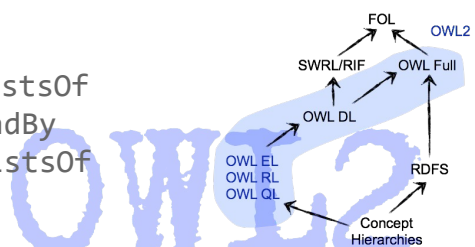
[1]

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`

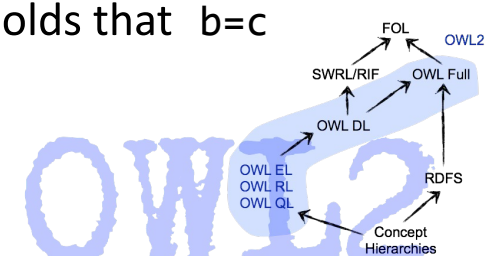
```
:isMadeOf a owl:ObjectProperty ;
       rdfs:subPropertyOf :consistsOf .

:reads a owl:ObjectProperty ;
       owl:inverseOf :isReadBy .

:composedOf a owl:ObjectProperty ;
       owl:equivalentProperty :consistsOf .
```

isMadeOf ⊑ consistsOf
reads- ≡ isReadBy
composedOf ≡ 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`

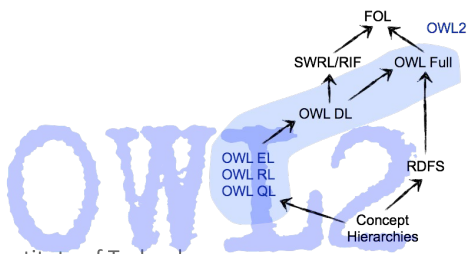


OWL Transitive Properties

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

:AnimalFarm a owl:Book ;
            :isPublishedBefore :NineteenEightyfour .

:BraveNewWorld a owl:Book ;
               :isPublishedBefore :AnimalFarm .
```

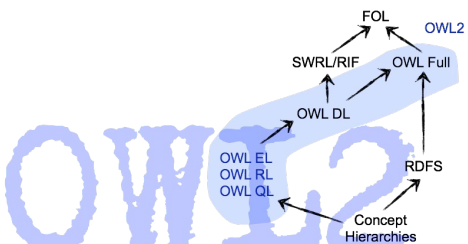
- via inference it can be entailed that

```
:BraveNewWorld :isPublishedBefore :NineteenEightyfour .
```



More Property Relationships

- **Asymmetric properties** via `owl:AsymmetricProperty`
  e.g.: if it holds that `isLeftOf(a,b)`
  then it is <u>not</u> 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≠y`



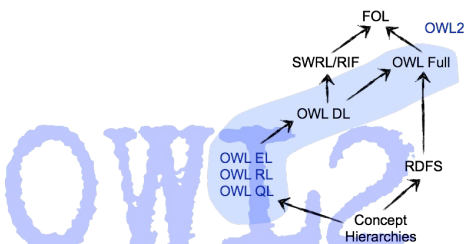
Disjunctive Properties

- 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 ) .
```

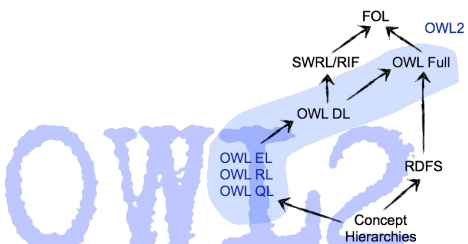


Negated Property Instantiations

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

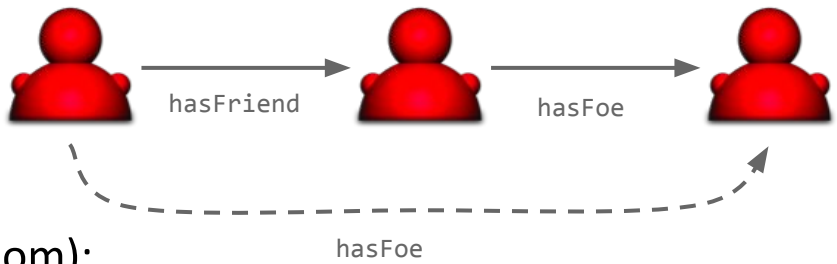
¬isBrother(GeorgeOrwell,AldousHuxley)

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

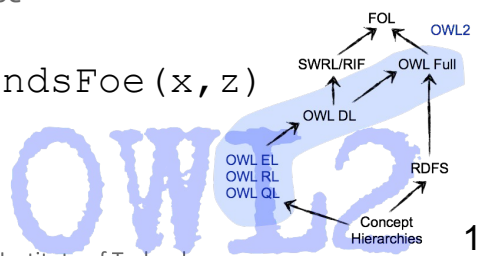


OWL Property Chaining

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

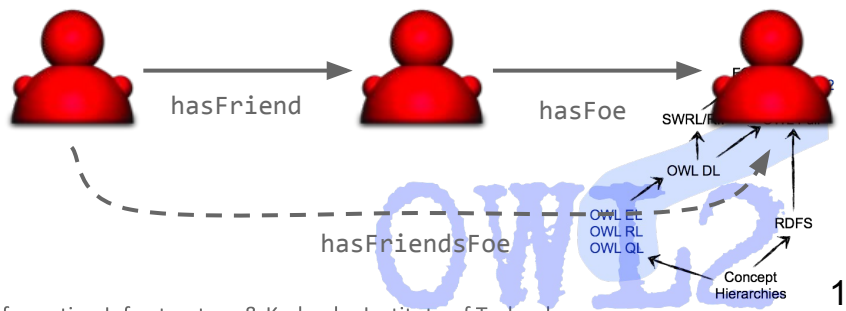


OWL Property Chaining

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

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

- Not allowed for datatype properties



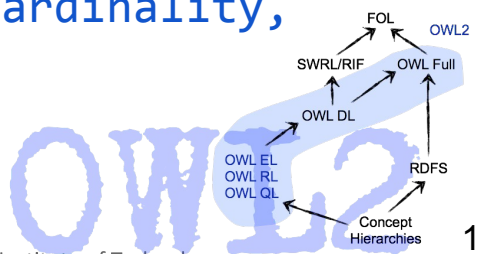
OWL Qualified Number Restrictions

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

$$\text{SuccessfulAuthor} \sqsubseteq \geq 1\ \text{notableWork.Bestseller}$$

```
: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

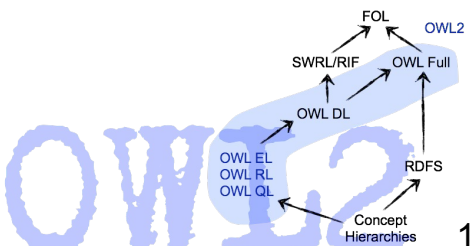


OWL Reflexive Property Restriction

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

$$\text{Philosopher} \sqsubseteq \text{knows.self}$$

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



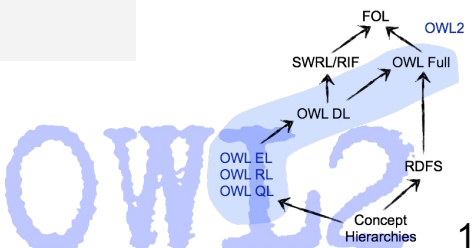
Beyond OWL

- 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"

- $\texttt{Squanderer} \sqsubseteq \texttt{Person}$
- $\texttt{Squanderer} \sqsubseteq \texttt{hasExpenses}.\top$
- $\texttt{Squanderer} \sqsubseteq \texttt{hasIncome}.\top$

?

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

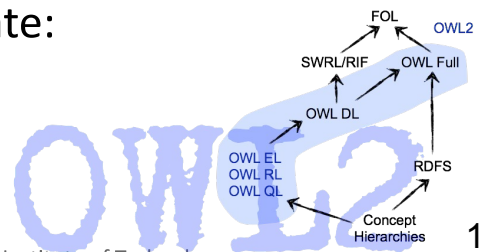


Rules Beyond OWL

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

$$\text{Person}(x) \land \text{hasIncome}(x,y) \land \text{hasExpenses}(x,z) \land (z > y) \rightarrow \text{Squanderer}(x)$$

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



Next Lecture...

Karlsruher Institut für Technologie
FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

Picture References:

- [1] Benjamin Nowack, *The Semantic Web - Not a Piece of cake…*, at bnode.org, 2009-07-08 , [CC BY 3.0]
  http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake
- [2] Rembrandt, The Anatomy Lesson of Dr. Nicolaes Tulp, 1632 [Public Domain]
  https://commons.wikimedia.org/wiki/File:Rembrandt_-_The_Anatomy_Lesson_of_Dr_Nicolaes_Tulp.jpg