Vorlesung

Webbasierte Informationssysteme

(CS4130)

Die Semantic Web-Ontologiesprachen RDFS und OWL

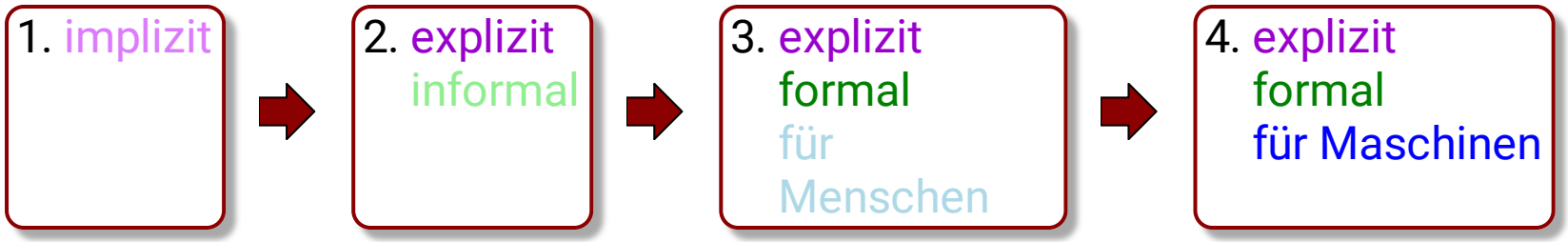
Professor Dr. rer. nat. habil. Sven Groppe
https://www.ifis.uni-luebeck.de/index.php?id=groppe

Chronologische Übersicht über die Themen

Überblick über die Vorlesung

- Einführung, u.a.
  - Definition von Ontologien
  - Sinn und Zweck von Ontologien
- Vorstellung der Semantic Web Ontologiesprachen
  - RDF Schema
    - Sprachkern
    - Axiomatische Formulierung der Semantik
  - Web Ontology Language (OWL)
    - Einführung in die Sprachkonstrukte

Semantik im Web – "The Semantic Continuum" (vgl. Uschold 2002)

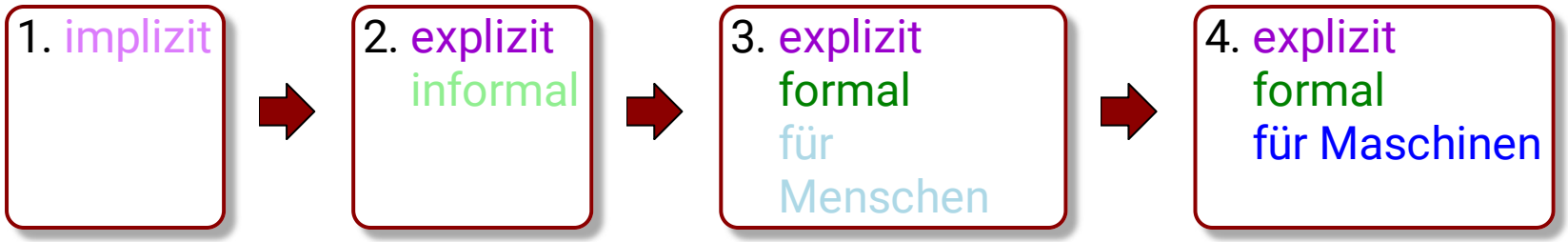


1. Implizite, kontextabhängige Semantik
   - Beispiel: „sprechende“ XML-Tags

2. Informale Sprache mit expliziter Semantik
   - Beispiele: Bedienungsanleitung, Glossar, Java-Doc

3. Formale Sprache mit expliziter Semantik
   - Beispiel: XML-Schema, Ontologie

Semantik im Web – "The Semantic Continuum" (vgl. Uschold 2002)



4. Modelltheoretische Semantik, axiomatische Semantik
   - schaffen die Voraussetzung, um das Wesen (die Semantik) von Schlussfolgerungsprozessen formal zu fassen
   - ➡ Schlussfolgerungsprozesse von Mensch und Maschine sind nicht unterscheidbar
   - ➡ die menschliche Semantik (des Schlussfolgerns) ist auf Rechner übertragbar
   - Beispiele: regelbasiertes System, Inferenzsystem für Ontologie

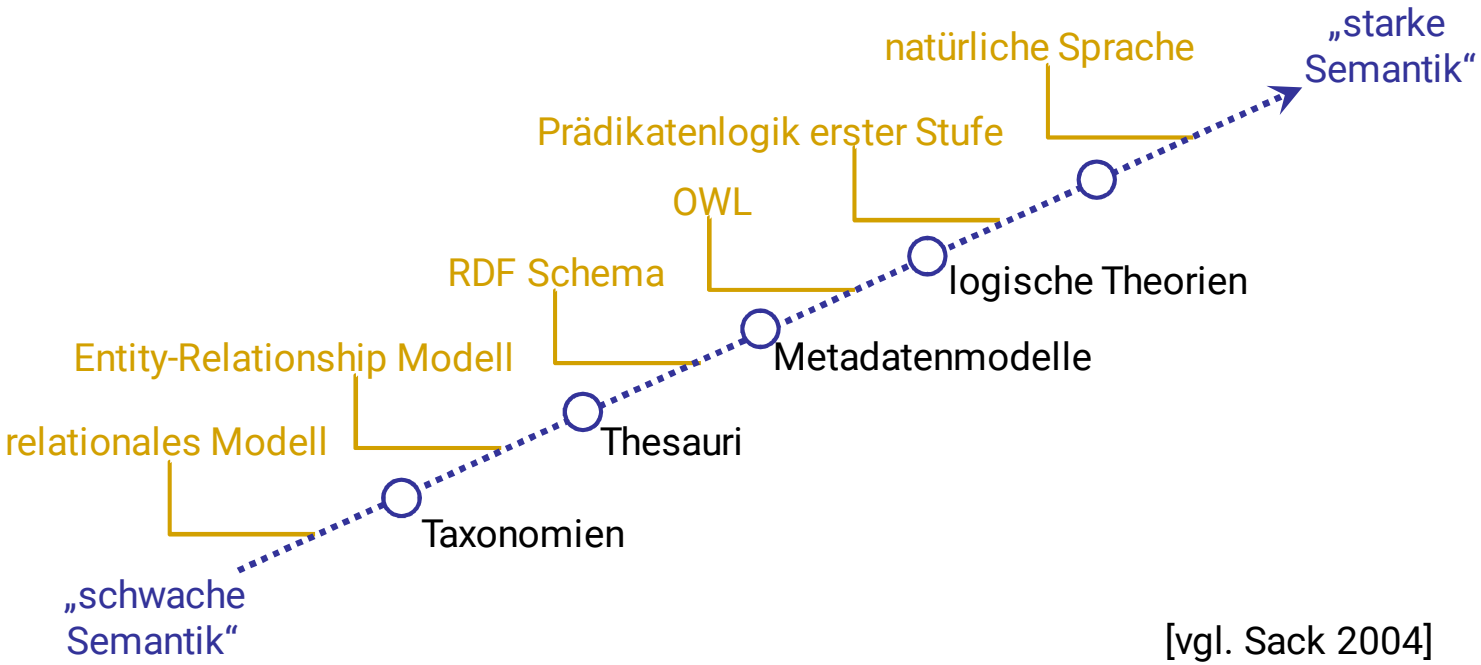
Definition von Ontologie

- Ontologie [griechisch] – die Lehre vom Seienden
- Definition nach Gruber 1993:

> An ontology is
> a formal specification
> of a shared
> conceptualization
> of a domain of interest.

➡ interpretierbar durch Maschinen
➡ gemeinsames Verständnis einer Gruppe
➡ es geht um Begriffsbildung
➡ bekannter (abgegrenzter) Gegenstandsbereich

Ontologien - Einteilung nach Ausdrucksstärke



[vgl. Sack 2004]

- Mächtigere Ausdrucksstärke zur Ontologiebeschreibung
  - ➡ stärkere intendierte Semantik
  - ➡ komplexere/langsamere Schlussfolgerungsprozesse im Worst-Case

Taxonomien

- Carl von Lineé um 1740:
  hierarchisches Klassifikationsschema für Tiere und Pflanzen
  ➡ Grundlage der modernen botanischen und zoologischen Taxonomie
- Systematik der Silbermöwe:

Reich (Regnum)	Tiere
Unterreich (Subregnum)	Vielzeller Metazoa
Abteilung (Divisio)	Vielzeller im engeren Sinne Eumetazoa
Stamm (Phylum)	Chordatiere Chordata
Unterstamm (Subphylum)	Wirbeltiere Vertebrata
Klasse (Classis)	Vögel Aves
Ordnung (Ordo)	Wat- und Möwenvögel Charadriiformes
Unterordnung (Subordo)	Möwenartige Lari
Familie (Familia)	Möwenvögel Laridae
Unterfamilie (Subfamilia)	Larinae
Gattung (Genus)	Möwen Larus
Art (Species)	Silbermöwe Larus argentatus

Sinn und Zweck von Ontologien

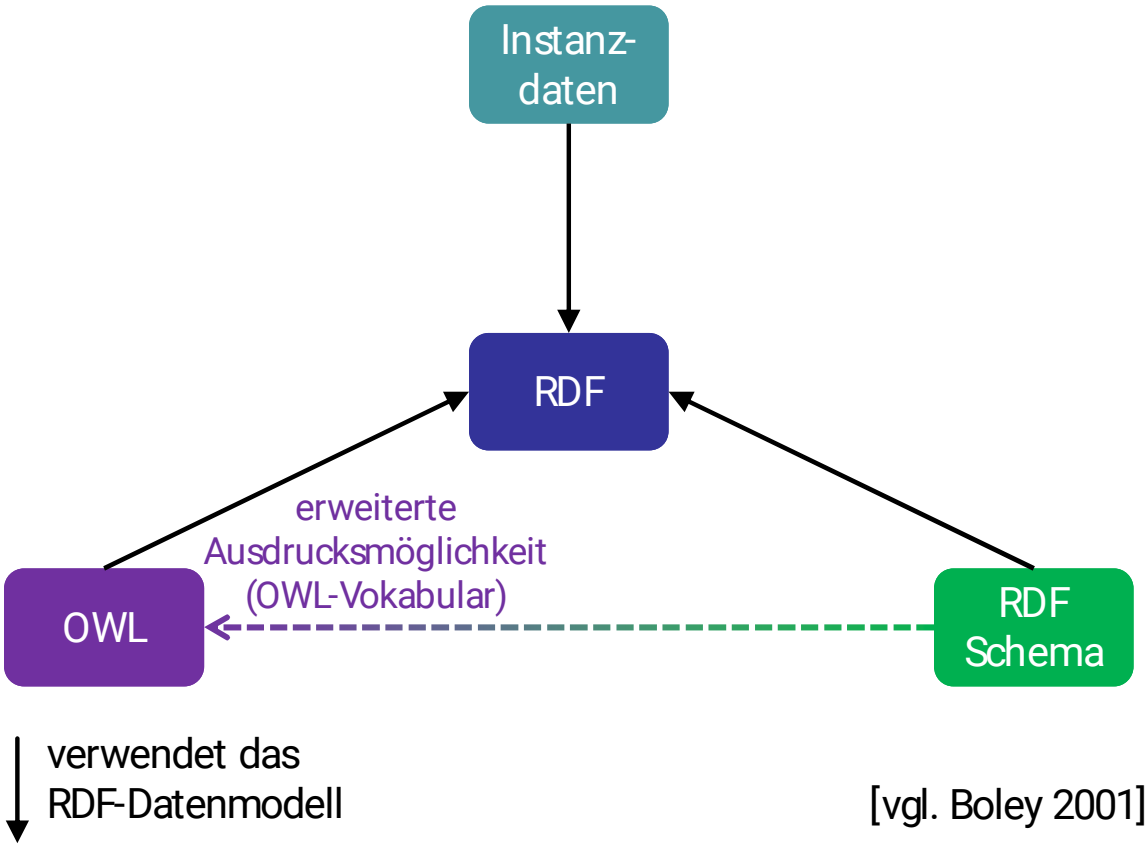
- Ontologien ermöglichen u.a.:
  - Realisierung „semantischer“ Suchfunktionen
    - Beispiele: automatische Spezialisierung, Generalisierung, Verfeinerung
  - Realisierung „intelligenter“ Dialoge mit Maschinen
    - Beispiel: Nachfragen bei Mehrdeutigkeiten
  - Interoperabilität beim Austausch von Daten mit Markup
- Elemente einer Ontologie-Sprache:
  - Klassen, auch Konzepte genannt
  - Eigenschaften der Klassen
  - Beziehungen zwischen Klassen und Eigenschaften
- Anwendungen basieren auf das Ziehen von logischen Schlussfolgerungen über (Teil-) Mengen von Konzepten

RDF Schema und OWL

- RDF Schema
  - Einfache Ontologiesprache zur Beschreibung von
    - Klassen- und Eigenschaftshierarchien, sowie
    - Definitions- und Wertebereiche von Eigenschaften

- Web Ontology Language (OWL)
  - Komplexe Ontologiesprache
  - Spezifikation von verschiedenen Teilmengen von OWL Full mit unterschiedlichen Berechnungskomplexitäten
  - In OWL 2.0 zusätzlich Spezifikation von einigen Profilen (ebenfalls Teilmengen von OWL Full) z.B. für
    - Speichern in relationalen Datenbanken und Anfrageumschreibung zu SQL,
    - Schlussfolgern mit regelbasierten Systemen,
    - etc.

Zusammenhang zwischen OWL und RDF/RDFS



[vgl. Boley 2001]

RDF Schema (RDFS): Eine einfache Ontologiesprache

- Kern von RDFS

RDFS Statement	Beschreibung
`Class1 rdf:type rdfs:Class.`	`Class1` ist eine Klasse
`Class2 rdfs:subClassOf Class1.`	`Class2` ist eine Unterklasse von `Class1`. Mehrfachvererbung erlaubt!
`A rdf:type Class1.`	`A` ist eine Instanz von `Class1`. Weitere Klassenzugehörigkeiten von `A` sind erlaubt!
`P1 rdf:type rdfs:Property.`	`P1` ist eine Property
`P2 rdfs:subPropertyOf P1.`	`P2` ist eine Unter-Property von `P1`
`A P1 "value".`	Verwendung der Property `P1`
`ex:mayDrive rdfs:domain ex:Person.`	`ex:mayDrive` hat den Definitionsbereich `ex:Person`
`ex:mayDrive rdfs:range ex:Vehicle.`	`ex:mayDrive` hat den Wertebereich `ex:Vehicle`

Entwicklung einer RDFS-Ontologie

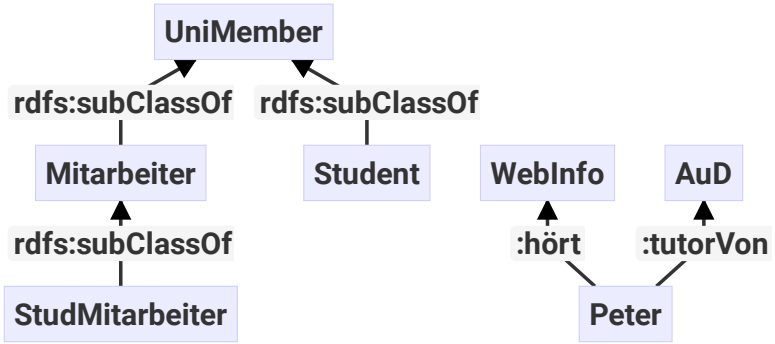
- Entwickle zu der folgenden Beschreibung eine RDFS-Ontologie und bestimme Instanzen:
  - *Universitätsangehörige* sind *Mitarbeiter* oder *Studenten*.
  - *Dozenten* gehören zu den *Mitarbeitern*.
  - *Dozenten* sind *Professoren*.
  - *Peter* ist ein *studentischer Mitarbeiter*, der ein *Tutor* von *AuD* ist und *Webbasierte Informationssysteme* hört.

Hinweis: Obige Angaben können (wie im richtigen Leben) unvollständig (oder sogar teilweise falsch) sein, d.h. evtl. muss sinnvoll ergänzt und korrigiert werden.

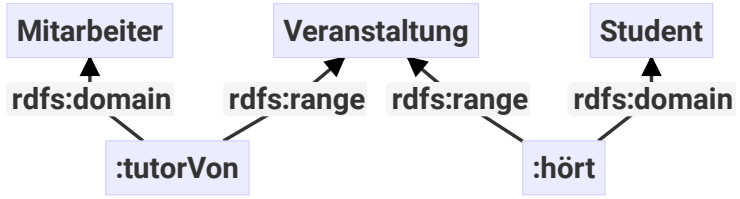
Axiomatische Formulierung der Semantik

- Gebe die Semantik und die folgerbaren Fakten zu folgender Ontologie an:

Klassenhierarchie und Instanzen:



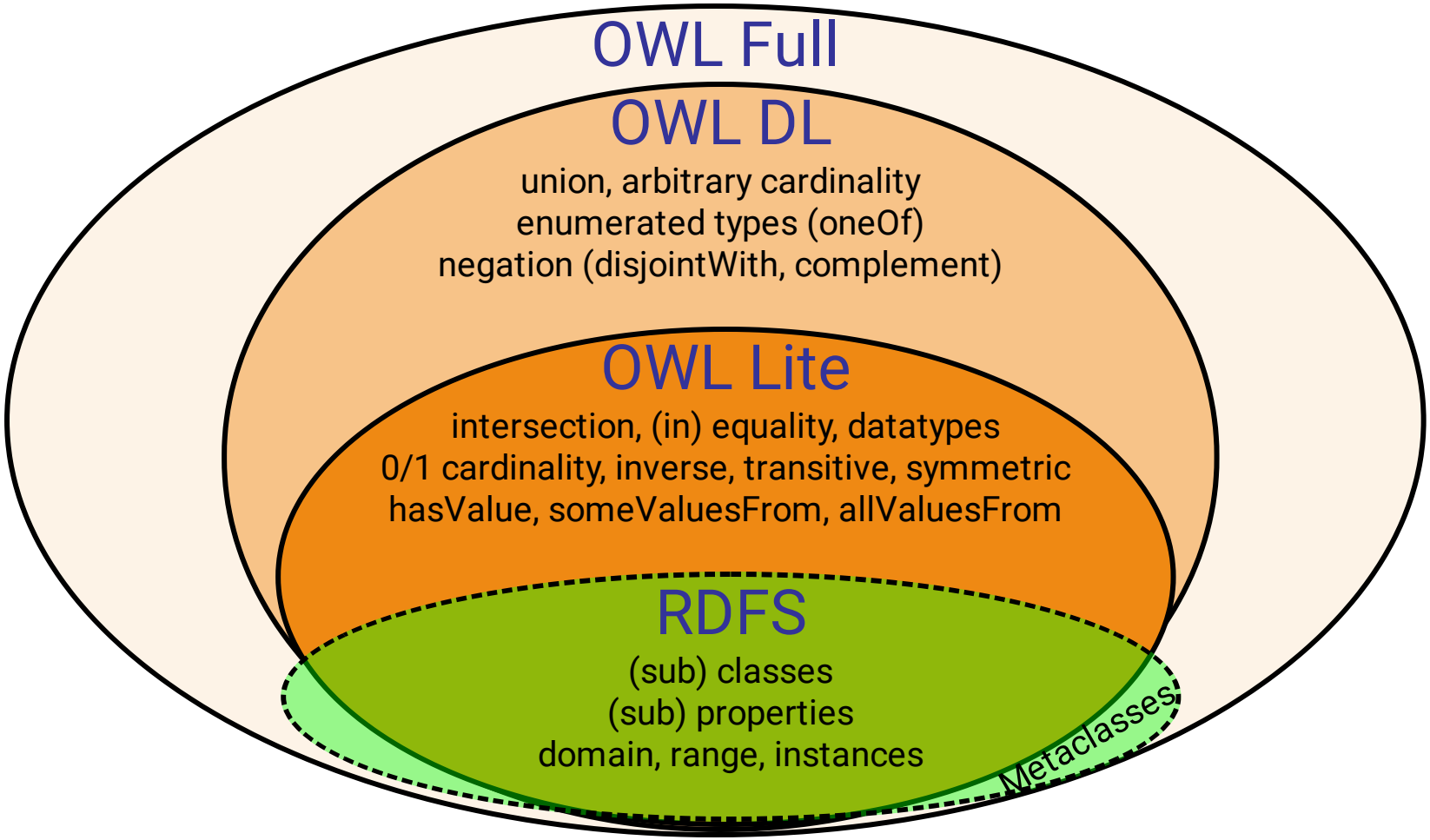
Properties:



Bemerkungen

- Axiomatische Formulierung der Semantik hier nur beispielhaft und längst nicht vollständig
- Inferenzregeln
  - Weitere Möglichkeit zur Spezifikation der Semantik von Ontologiesprachen:
    Verwendung von Regeln (bestehend aus Prämisse und Konklusion) für die formale Beschreibung der Herleitung neuen Wissens auf Grund einer Ontologie
- Beide Beschreibungsarten der Semantik prinzipiell auch für OWL möglich (Inferenzregeln nur für das Fragment OWL2RL)

OWL 1.0 (Lite, DL, Full) und RDF Schema



OWL – Ausgewählte Sprachkonstrukte

- Instanzen
  - Gleichheit, Ungleichheit, Negative Aussage
- Klassen
  - Äquivalenz, Disjunktheit, Durchschnitt, Vereinigung, Komplement, Definition durch Existenz-/Allquantor/ Aufzählung
- Eigenschaften
  - Kardinalitäten, invers, (a)symmetrisch, (ir)reflexiv, disjunkt, (invers-) funktional, transitiv

(Un-) Gleichheit, Negative Aussage

- Gleichheit von Instanzen

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

Extensive Nutzung in Linked Data für Verweise zwischen Datensätzen

- Ungleichheit von Instanzen

```
:John owl:differentFrom :Bill.
```

- Negative Aussage

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

`:Mary` ist **nicht** die Frau von `:Bill`

Klassen - Mengenoperationen 1/2

- Vereinigung von Klassen

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

- Durchschnitt von Klassen

```
:Mother owl:equivalentClass [
        rdf:type owl:Class;
        owl:intersectionOf ( :Woman :Parent )].
```

Klassen - Mengenoperationen 2/2

- Äquivalente Klassen

```
:Person owl:equivalentClass :Human.
```

- Disjunktheit von Klassen

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

- Komplement von Klassen

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

Quantoren bei Properties

- Existenzquantor

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

Hier: Definition von Eltern über die Existenz einer :hasChild Property.

- Allquantor

```
:HappyPerson rdf:type owl:Class ;
             owl:equivalentClass [
                 rdf:type          owl:Restriction;
                 owl:onProperty    :hasChild;
                 owl:allValuesFrom :HappyPerson] .
```

Hier: Eltern sind glücklich, wenn alle ihre Kinder glücklich sind.

Definition von Klassen über Aufzählung und Propertywerte

- Klassendefinition durch Aufzählung der Instanzen

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

Hier: Meine Geburtstagsgäste sind `:Bill`, `:John` und `:Mary`.

- Klassendefinition über Propertywerte

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

Hier: John's Kinder erhalten eine eigene Klasse.

Eigenschaften von Properties

- Kardinalitäten (hier: ohne Einschränkung der Klassenzugehörigkeit)

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

- Inverse Properties

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

Damit: `:John :hasChild :Bo.` $\Leftrightarrow$ `:Bo :hasParent :John.`

Eigenschaften von Properties

- ## Symmetrische Properties

```
:hasSpouse rdf:type owl:SymmetricProperty .
```

Spouse = Ehepartner

Damit: `:John :hasSpouse :Mary.` ⇔ `:Mary :hasSpouse :John.`

- ## Asymmetrische Properties

```
:hasChild rdf:type owl:AsymmetricProperty .
```

Damit: `:John :hasChild :Bo.` **niemals** ~~`:Bo :hasChild :John.`~~

Eigenschaften von Properties

- ## Disjunkte Properties

```
:hasParent owl:propertyDisjointWith :hasSpouse.
```

Hier: Der Ehepartner kann nicht gleichzeitig ein Elternteil sein!

- ## Reflexive Properties

```
:hasRelative rdf:type owl:ReflexiveProperty.
```

Hier: Jeder ist mit sich selbst verwandt.

- ## Irreflexive Properties

```
:parentOf rdf:type owl:IrreflexiveProperty.
```

Hier: Niemand ist von sich selbst ein Elternteil!

Eigenschaften von Properties

- ## Funktionale Properties

```
:hasHusband rdf:type owl:FunctionalProperty.
```

Hier: Jemand hat höchstens einen Ehegatten!

- ## Invers-Funktionale Properties

```
:hasHusband rdf:type owl:InverseFunctionalProperty.
```

Hier: Man kann höchstens von einer anderen der Ehegatte sein!

- ## Transitive Properties

```
:hasAncestor rdf:type owl:TransitiveProperty.
```

Hier: Die Vorfahren der Vorfahren sind auch Vorfahren wie auch deren Vorfahren usw.

Zusammenfassung

- Definition von Ontologie
- Anwendungsgebiete von Ontologien
- Einführung in die Ontologiesprachen RDF Schema und OWL
  - Axiomatische Formulierung der Semantik
  - Nicht alle Sprachkonstrukte behandelt
  - ➡ Hinweis auf die Veranstaltung Semantic Web (Teil des Vertiefungsmoduls Datenmanagement)

In den Übungen:
Design (und Test) einer OWL-Ontologie!