

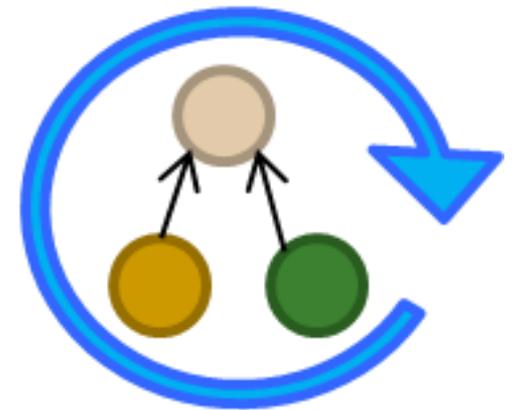
# Ontologie-Management

## *Kapitel 2: Ontologiesprachen*

Wintersemester 2013/14

Anika Groß

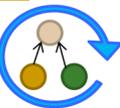
Universität Leipzig, Institut für Informatik  
Abteilung Datenbanken  
<http://dbs.uni-leipzig.de>



*Die Folien zur Vorlesung „Ontologie Management“  
wurden von Dr. Michael Hartung erstellt.*

# Inhalt

- Erinnerung Ontologie / Ontologiemodell
- Ontologiesprachen
  - Framebasiert: F-Logic
  - Semantic Web: RDF / RDFS / OWL
  - Open Biomedical Ontologies: OBO



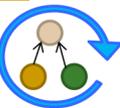
# Erinnerung Ontologie

“An ontology is an **explicit, formal specification of a shared conceptualization**. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what “exists” is exactly that which can be” represented. *(Thomas R. Gruber, 1993)*



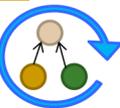
„explizite, formale Spezifikation einer gemeinsamen Konzeptualisierung“

- **Konzeptualisierung** abstraktes Modell (Domäne, identifizierte relevante Begriffe, Beziehungen)
- **Explizit** Bedeutungen aller Begriffe definiert
- **Formal** maschinenverstehbar
- **Gemeinsam** Konsens bzgl. Ontologie



# Erinnerung Ontologiemodell

- **Ontologie  $O = (C, A, R)$** 
  - Konzepte  $c \in C$  (Identifizierung über *accession number*)
  - Attribute  $a = (a_{concept}, a_{name}, a_{value}) \in A$
  - Relationen  $r = (r_{source}, r_{type}, r_{target}) \in R$
- **Beispiel – Blutgerinnung in GO Biologische Prozesse (GO:0007596)**
  - *name*: blood coagulation
  - *synonym*: blood clotting
  - *obsolete*: false
  - *definition*: „The sequential process ...“
  - *is\_a*: GO:0050817, GO:0007599
  - *part\_of*: GO:0042060



# Ontologiesprachen - Überblick

## ■ F-Logic

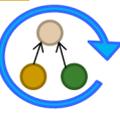
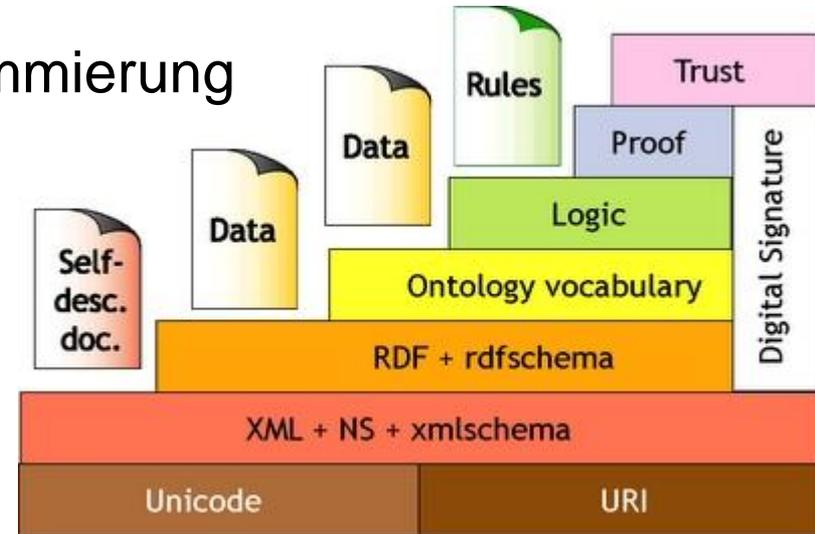
- Objektorientierte logische Programmierung

## ■ RDF / RDFS / OWL

- W3C Standard
- Basierend auf Description Logic

## ■ OBO

- Open Biomedical Ontologies
- Einheitliche Modellierung biomedizinischer Ontologien



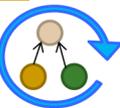
# F-Logic (Frame-Logic)

## ■ Motivation

- ❑ Beschreibungssprachen wie OWL oft „eigenschaftszentriert“
  - Relationen (Rollen) und Klassen als Grundlage zur Klassifizierung von Instanzen
- ❑ Programmierung typischerweise objektorientiert
  - Relationen (Eigenschaften) sind Klassen zugeordnet, Instanzen als komplex strukturierte Objekte

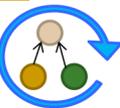
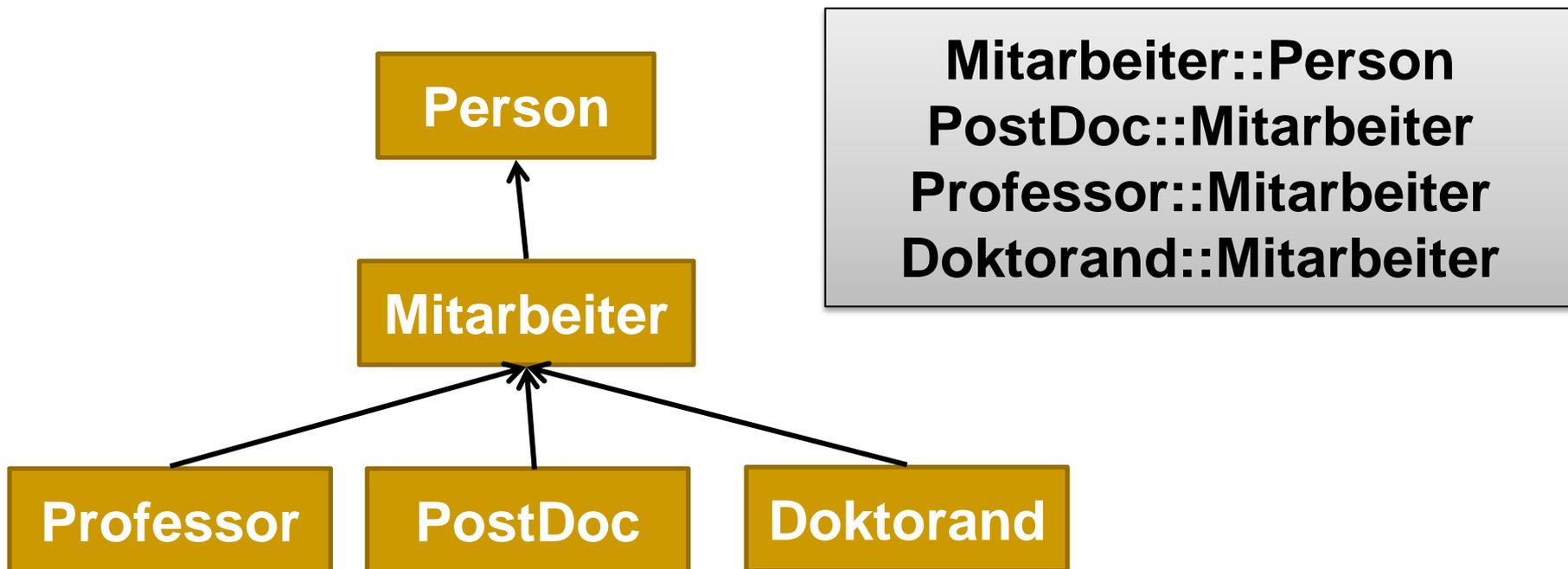
## ■ Eigenschaften

- ❑ Selbstbeschreibendes OO-Datenmodell (M. Kifer, G. Lausen, 1989/1995)
- ❑ Volle Objektorientierung (Klassenhierarchie, Vererbung)
- ❑ Deduktive Sprache (Prolog-artig)



# F-Logic – Definition von Klassen

- Klassen und Subklassen-Beziehungen
  - Eindeutige Namen für Klassen: *class*
  - *class1::class2* zur Beschreibung einer Subklassen-Beziehung zwischen zwei Klassen *class1* und *class2*



# F-Logic – Definition von Relationen

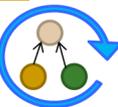
- Definition von Relationen zwischen Klassen
  - Beschreibung der Klassen im Detail
  - Unterscheidung zwischen ein- und mehrwertigen Relationen
    - $class1[relation=>class2]$  (einwertig =>)
    - $class1[relation=>>class2]$  (mehrwertig =>>)



**Doktorand[wirdBetreutVon=>Professor]**



**Mitarbeiter[verantwortlichFür=>>Lehrveranstaltung]**



# F-Logic – Definition von Instanzen

- Zuordnung von Instanzen zu Klassen (Instanziierung)

- *object: class*

```
Rahm:Professor .  
Hartung:PostDoc .  
Kolb:Doktorand .
```

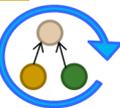
- Zuordnung von Werten für Relationen

- Unterscheidung zwischen ein- bzw. mehrwertig (-> bzw. ->>)

```
Kolb[wirdBetreutVon->Rahm] .
```

```
Rahm[verantwortlichFür->>{DBS1,IDBS1}] .
```

```
Hartung[verantwortlichFür->>OnMa] .
```



# F-Logic – Atome vs. Moleküle

- Atom beschreibt jeweils eine einzelne Aussage (Fakt)

**Instanziierung: Kolb:Doktorand .**

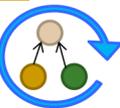
**1. Beziehung: Kolb[wirdBetreutVon->Rahm] .**

**2. Beziehung: Kolb[verantwortlichFür->>DWH-Prak] .**

- Moleküle fassen mehrere Fakten zusammen

**Kolb:Doktorand[wirdBetreutVon->Rahm;  
verantwortlichFür->>DWH-Prak] .**

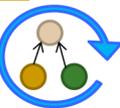
- Zerlegung von Molekülen in Atome möglich



# F-Logic – Einbettung (Nesting)

- Objektbeschreibungen können auch geschachtelt werden

```
Kolb:Doktorand[wirdBetreutVon->Rahm:Professor  
[verantwortlichFür->>{DBS1,IDBS1}];  
verantwortlichFür->>DWH-Prak] .
```



# F-Logic – Regeln

- Regeln bestehen aus Kopf und Rumpf (Bedingungen)
  - Regeln sind Implikationen (Implikationspfeil  $\leftarrow$ )
  - Beliebige logische Ausdrücke im Rumpf
  - Quantifizierung von Variablen mittels FORALL bzw. EXISTS

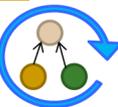
**FORALL X,Y X[istDoktorvaterVon->>Y]  
 $\leftarrow$   
Y:Doktorand[wirdBetreutVon->X] .**

*„Für alle X und Y gilt: X ist der Doktorvater von Y, wenn Y ein Doktorand ist und von X betreut wird.“*

Variablen

Rumpf  
(Bedingungen)

Regelkopf



# F-Logic – Anfragen

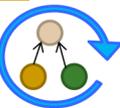
- Regeln ohne Kopf
  - Ergebnis sind Variablenbindungen, die die gegebenen Bedingungen erfüllen

**FORALL X**

**<-**

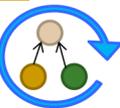
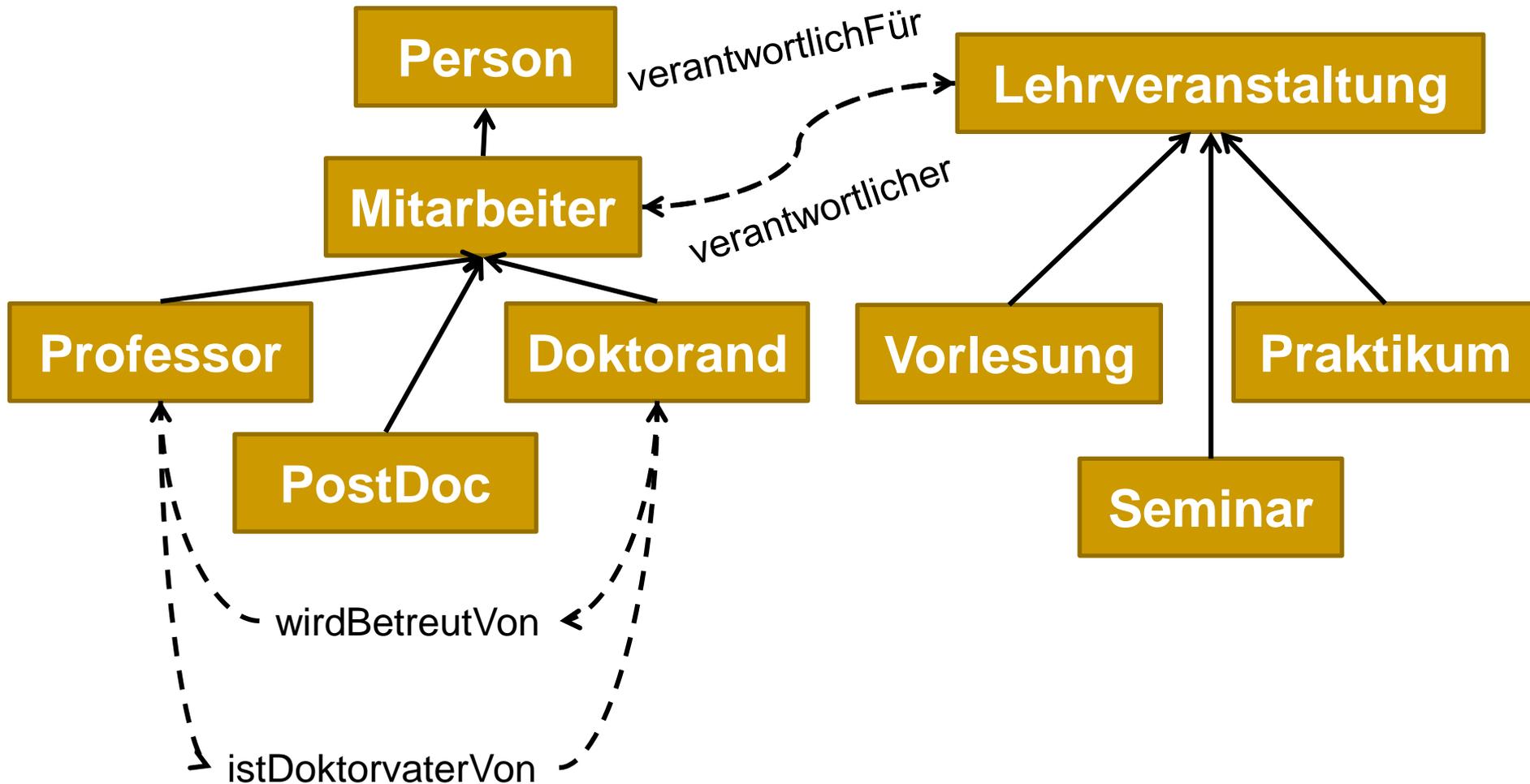
**EXISTS Y X:Doktorand[istVerantwortlichFür->>Y] .**

*„Finde alle Doktoranden die mindestens eine Lehrveranstaltung betreuen, d.h. Doktoranden die auch in der Lehre tätig sind.“*



# F-Logic – Gesamtbeispiel

- „Ontologie“ für Lehrstühle



# F-Logic – Gesamtbeispiel

- Klassen und Subklassen

```
Mitarbeiter::Person .  
Professor::Mitarbeiter .  
PostDoc::Mitarbeiter .  
Doktorand::Mitarbeiter .
```

```
Vorlesung::Lehrveranstaltung .  
Praktikum::Lehrveranstaltung .  
Seminar::Lehrveranstaltung .
```

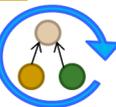
- Relationen

```
Doktorand[wirdBetreutVon=>Professor] .  
Professor[istDoktorvaterVon=>>Doktorand] .  
Mitarbeiter[verantwortlichFür=>>Lehrveranstaltung] .  
Lehrveranstaltung[verantwortlicher=>>Mitarbeiter] .
```

- Regeln

```
FORALL X,Y X[istDoktorvaterVon->>Y]  
<-  
Y:Doktorand[wirdBetreutVon->X] .
```

Weitere Regeln?

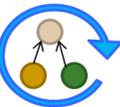


# F-Logic – Gesamtbeispiel

- Abteilung Datenbanken im WS 2012/13

**Rahm:Professor[verantwortlichFür->>{DBS1:Vorlesung, IDBS1:Vorlesung, Problemseminar:Seminar}; istDoktorvaterVon->>{Kolb:Doktorand [verantwortlichFür->>DWH-Prak:Praktikum], Arnold:Doktorand, Groß:Doktorand, Köpcke:Doktorand, Maßmann:Doktorand}] .**

**Hartung:PostDoc[verantwortlichFür->>OnMa:Vorlesung] .**



# Vor- und Nachteile F-Logic

## ■ Vorteile

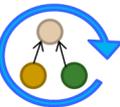
- ❑ Große Ausdrucksstärke (Regeln, ...)
- ❑ Objektorientiert
- ❑ Leistungsstark im Umgang mit Instanzen

## ■ Nachteile

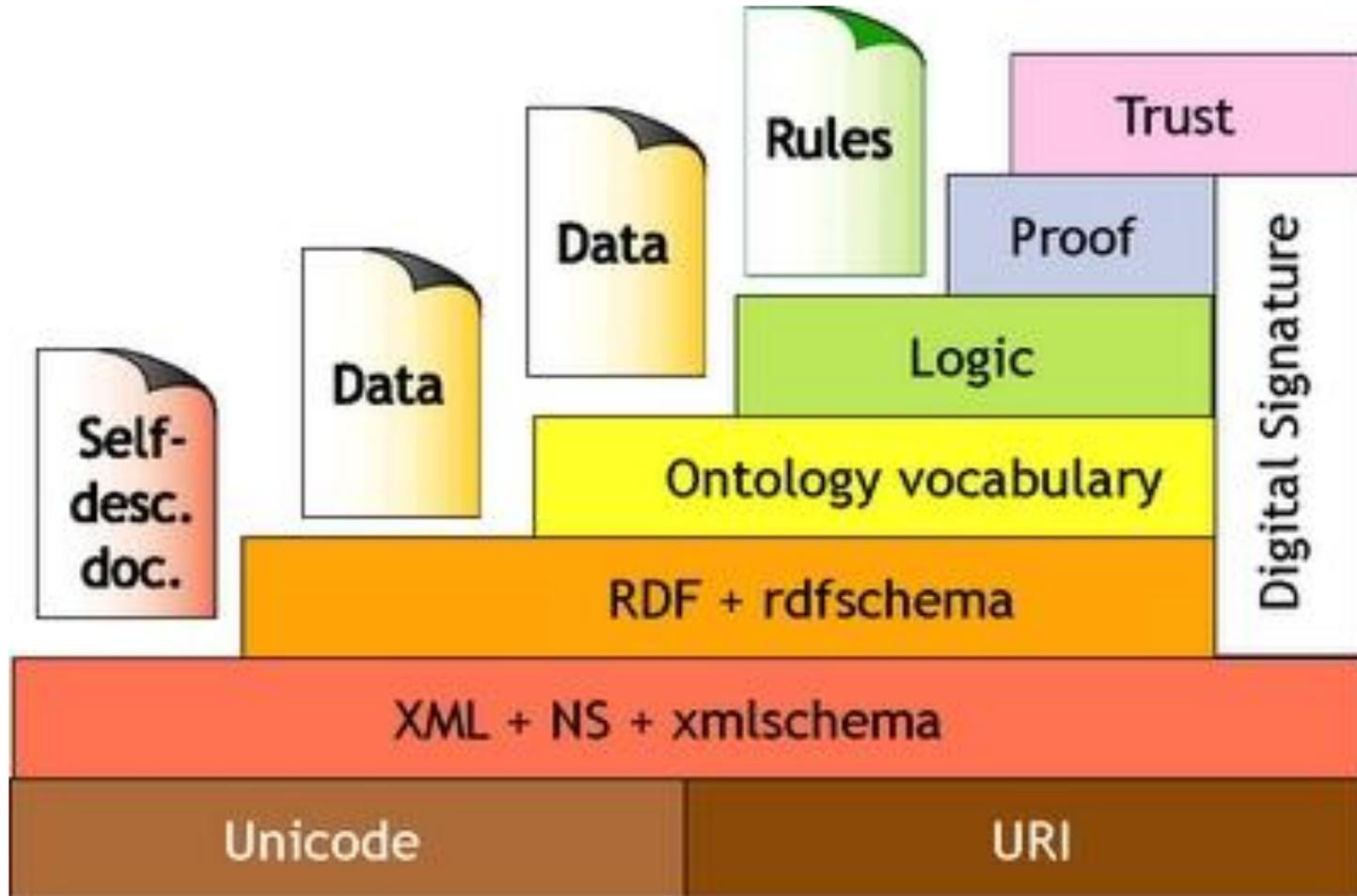
- ❑ Keine klassische Logik
- ❑ Verschiedene mögliche Semantiken
- ❑ Hochgradig unentscheidbar

## ■ Praktische Umsetzung

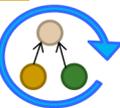
- ❑ Weitere Features: Pfadausdrücke für Eigenschaften, Anbindung DB, Namensräume, ...
- ❑ Systeme: OntoBroker, OntoStudio (ontoprise GmbH)
- ❑ Praxisorientierte Algorithmen



# Semantic Web - Architektur

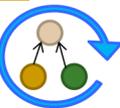


T. Berners-Lee: <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>



# W3C – Standards im Bereich Ontologien

- **XML (eXtensible Markup Language)**
  - ❑ Strukturierte Dokumente (Datenaustausch)
  - ❑ Keine explizite Bedeutung der Dokumentinhalte
- **RDF (Resource Description Framework)**
  - ❑ Beschreibung von Web-Ressourcen
  - ❑ Datenmodell für Objekte (Ressourcen) und deren Beziehungen untereinander (Properties)
- **RDF/S (RDF Schema)**
  - ❑ RDF Erweiterung zur Definition von Klassen, Beziehungen und Klassenhierarchien
- **OWL (Web Ontology Language)**
  - ❑ RDF/S Erweiterung zur detaillierten Beschreibung von Klassen, z.B. Kardinalitäten, Disjunktheit, ...



# RDF - Resource Description Framework

## ■ Resource

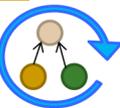
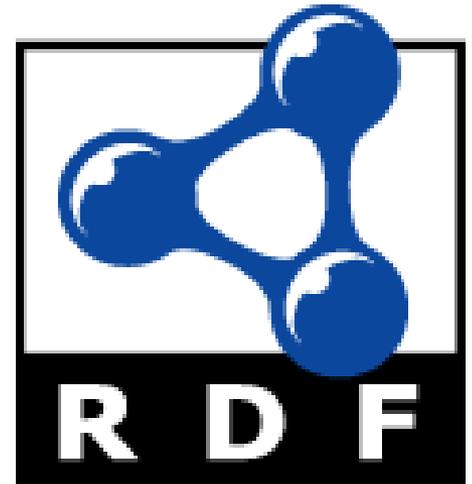
- ❑ Kann prinzipiell alles sein
- ❑ Bedingung: eindeutige Identifizierung/Referenz
- ❑ Typischerweise über URI

## ■ Description

- ❑ Beschreibungen der Ressourcen
- ❑ Mittels Beziehungen zwischen Ressourcen
- ❑ Darstellung als Graph

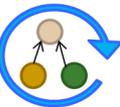
## ■ Framework

- ❑ Kombination aus webbasierten Protokollen (URI, HTTP, XML, ...)
- ❑ Definition erlaubter Beziehungen



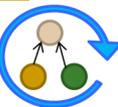
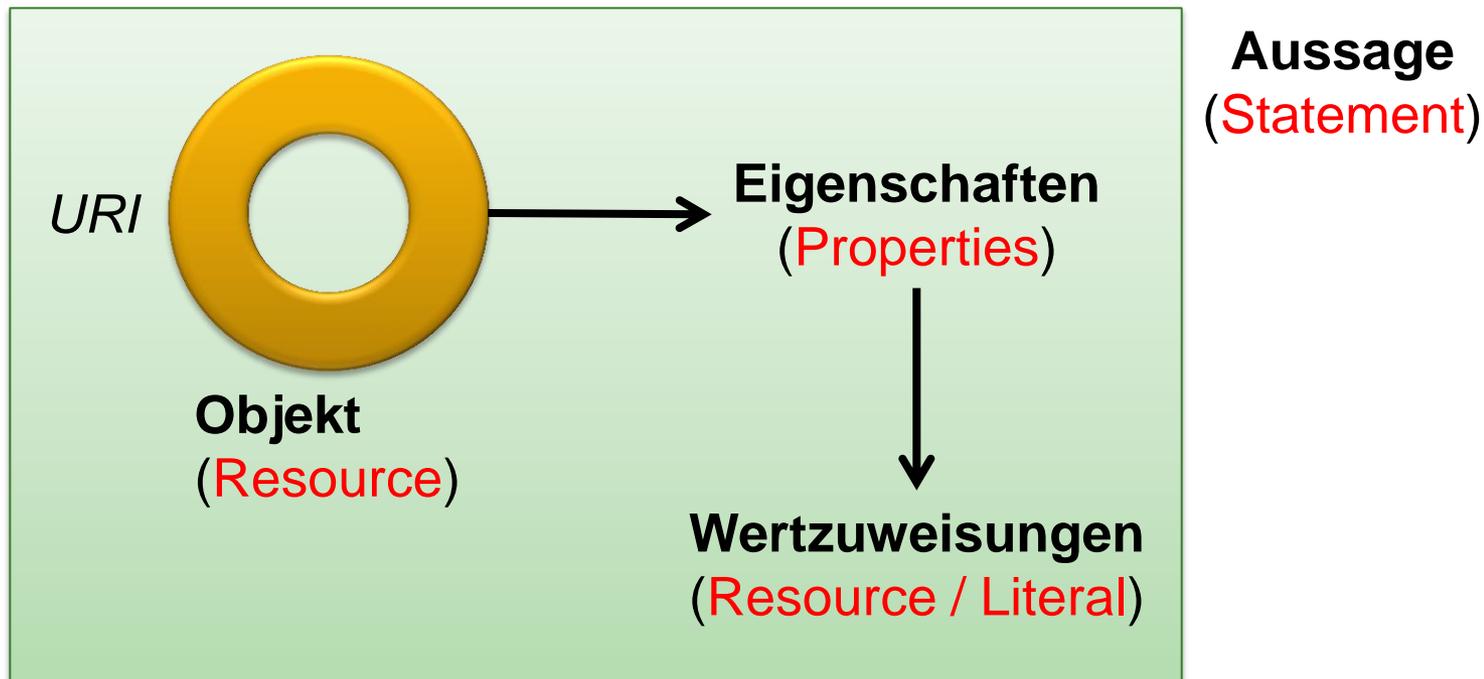
# RDF - Resource Description Framework

- Ursprünglich entwickelt zur Angabe von Metadaten für Web-Ressourcen
  - 1995-1997: proprietäres Meta Content Framework (Netscape)
  - 1997 RDF als allgemeine Sprachdefinition für Metadaten, W3C Draft
  - 1998 erste RDF W3C Recommendation (<http://www.w3c.org/RDF>)
  - 2004 überarbeitete RDF W3C Recommendation
- Festlegung einer XML-Untermenge mit fest vorgeschriebener Semantik  
(machine readable → machine understandable)
- RDF ist geeignet zur Beschreibung aller möglichen Web-Ressourcen
- Mit RDF soll ein möglichst hohes Maß an Interoperabilität ermöglicht werden



# RDF - Resource Description Framework

- Definiert Datenmodell zur Beschreibung maschinenverarbeitbarer Semantik von Daten
- Erlaubt Assoziation einfacher Semantik mit verwendeten Elementen



# RDF - Resource Description Framework

## ■ Ressourcen

- Dinge/Objekte, die über eine **URI** adressiert werden können

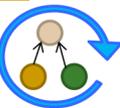
## ■ Properties

- Attribute/Eigenschaften zur Beschreibung der Ressource

## ■ Statements (RDF-Tripel)

- Ressource + Property + zugehöriger Wert

*T (subject S, property P, object O)*



# RDF Graphen

## Bestandteile

### ■ URI

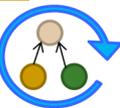
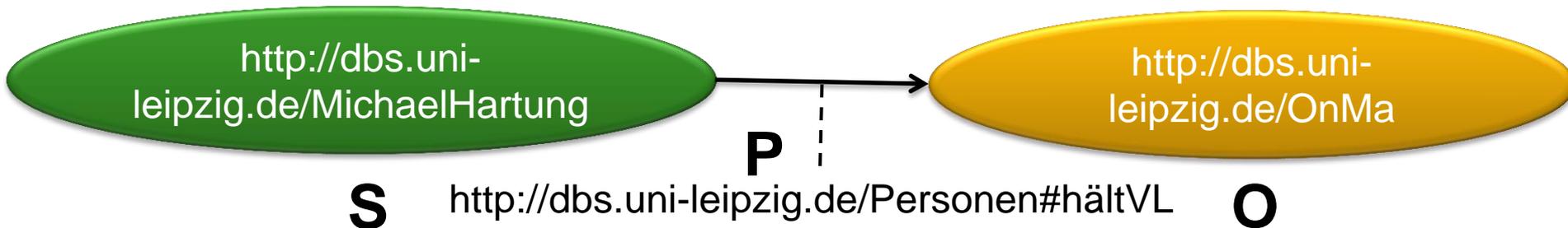
- Eindeutige Referenzierung der Ressourcen

### ■ Literale

- Datenwerte, welche keine separate Existenz aufweisen
- Zeichenketten, Interpretation über Datentyp

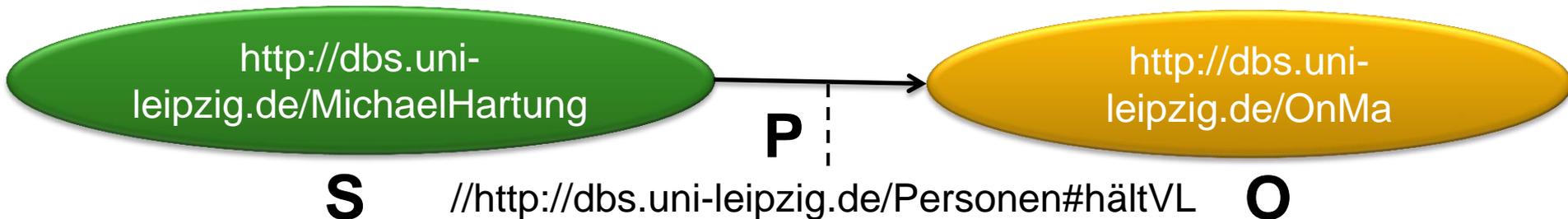
### ■ Leere Knoten

- Erlauben Existenzaussagen über ein Individuum mit gewissen Eigenschaften, ohne dieses zu benennen



# RDF Darstellungsformen

## ■ Knoten-Kante-Knoten Tripel



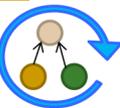
## ■ Notation3 (N3)

### □ Auflistung der Tripel

- { `http://dbs.uni-leipzig.de/MichaelHartung,`  
`http://dbs.uni-leipzig.de/Personen#hältVL`  
`http://dbs.uni-leipzig.de/OnMa` }

## ■ Turtle (Terse RDF Triple Language)

### □ N3 Erweiterung

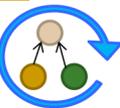


# RDF Darstellungsformen

## Turtle (Terse RDF Triple Language)

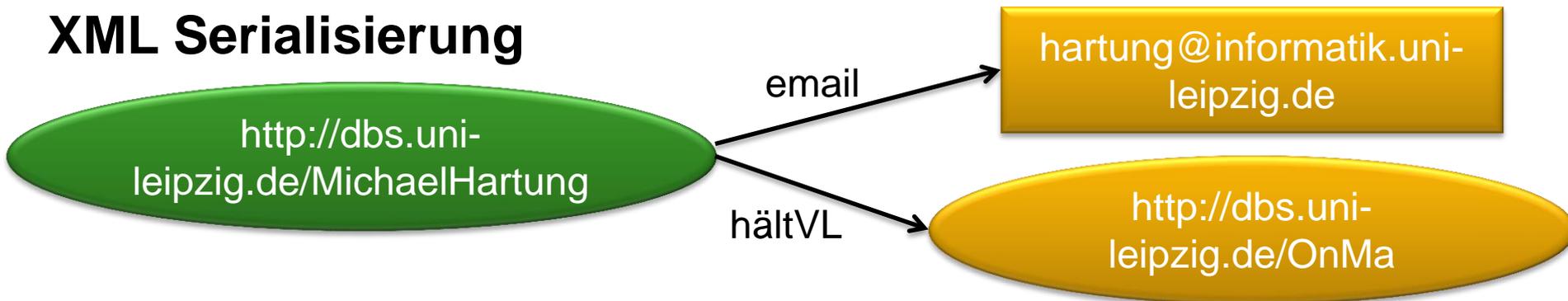
- Erweiterung von N3
- URIs in spitzen Klammern
- Literale in Anführungszeichen
- Jedes Tripel durch Punkt abgeschlossen
- **Beispiel:**

```
<http://dbs.uni-leipzig.de/MichaelHartung>  
<http://dbs.uni-leipzig.de/Personen#hältVL>  
<http://dbs.uni-leipzig.de/OnMa>.  
  
<http://dbs.uni-leipzig.de/MichaelHartung>  
<http://dbs.uni-leipzig.de/Personen#email>  
"hartung@informatik.uni-leipzig.de".
```

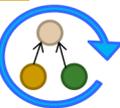


# RDF Darstellungsformen

## XML Serialisierung

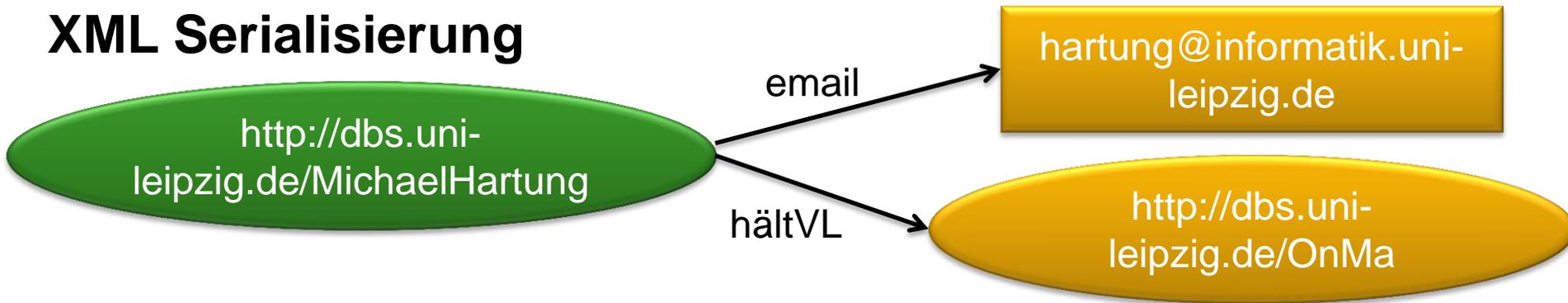


```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pers="http://dbs.uni-leipzig.de/Personen#">
<rdf:Description rdf:about="http://dbs.uni-leipzig.de/MichaelHartung">
  <pers:email>hartung@informatik.uni-leipzig.de</pers:email>
</rdf:Description>
<rdf:Description rdf:about="http://dbs.uni-leipzig.de/MichaelHartung">
  <pers:hältVL>
    <rdf:Description rdf:about="http://dbs.uni-leipzig.de/OnMa"></rdf:Description>
  </pers:hältVL>
</rdf:Description>
</rdf:RDF>
```

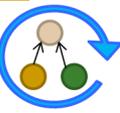


# RDF Darstellungsformen

## XML Serialisierung

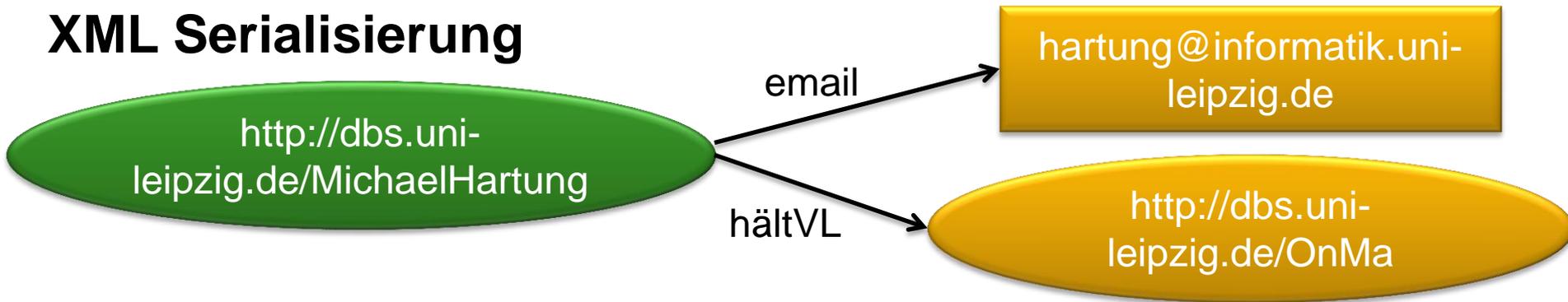


```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pers="http://dbs.uni-leipzig.de/Personen#">
  <rdf:Description rdf:about="http://dbs.uni-leipzig.de/MichaelHartung">
    <pers:email>hartung@informatik.uni-leipzig.de</pers:email>
  </rdf:Description>
  <rdf:Description rdf:about="http://dbs.uni-leipzig.de/MichaelHartung">
    <pers:hältVL rdf:resource="http://dbs.uni-leipzig.de/OnMa" />
  </rdf:Description>
</rdf:RDF>
```

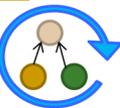


# RDF Darstellungsformen

## XML Serialisierung



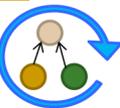
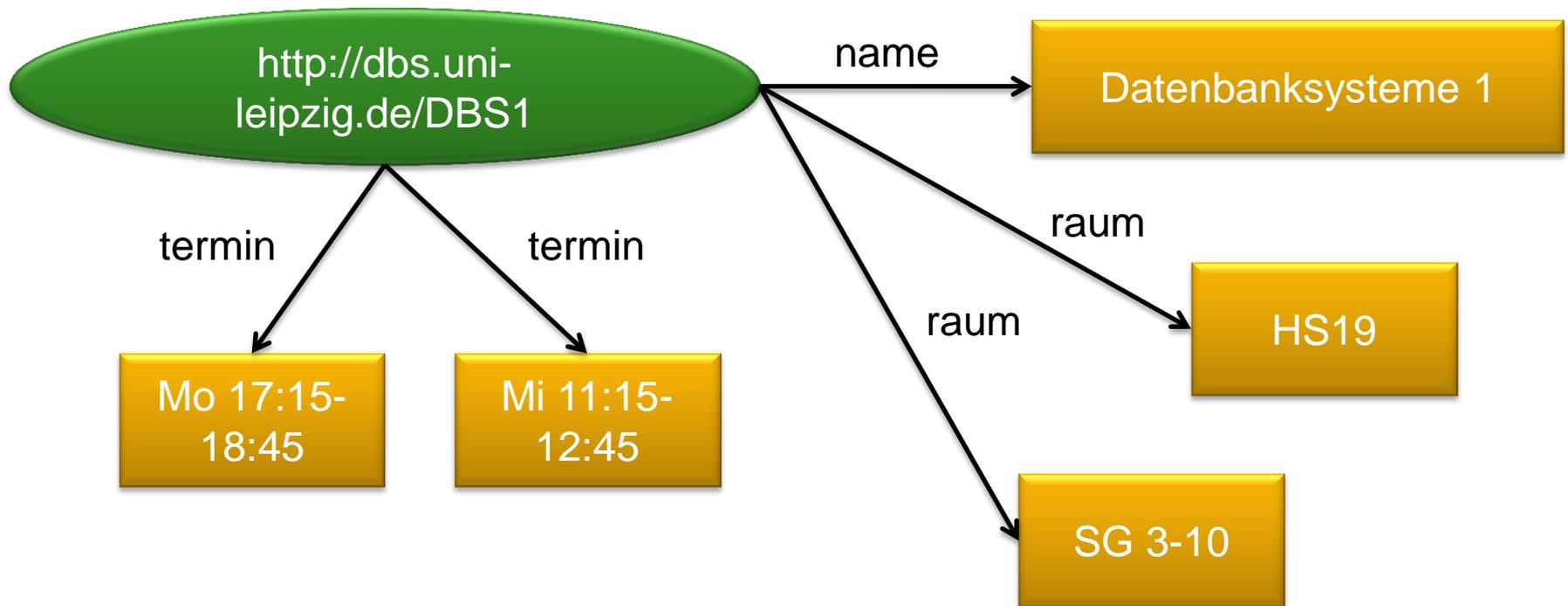
```
<xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pers="http://dbs.uni-leipzig.de/Personen#">
<rdf:Description rdf:about="http://dbs.uni-leipzig.de/MichaelHartung"
  pers:email="hartung@informatik.uni-leipzig.de">
  <pers:hältVL rdf:resource="http://dbs.uni-leipzig.de/OnMa" />
</rdf:Description>
</rdf:RDF>
```



# Mehrwertige Beziehungen und leere Knoten

## Beispiel

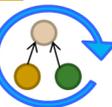
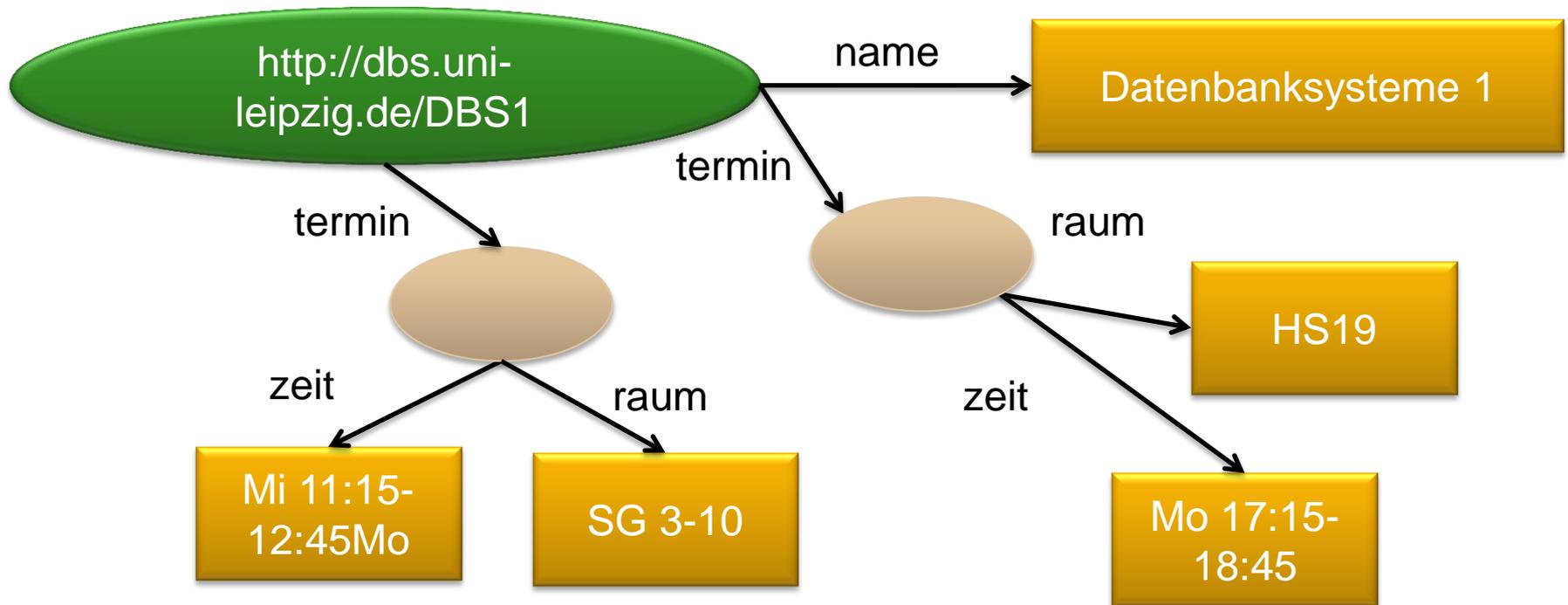
- Lehrveranstaltung mit mehreren Terminen in unterschiedlichen Räumen
- Modellierung in RDF?



# Mehrwertige Beziehungen und leere Knoten

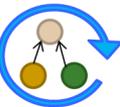
## Lösung

- Leere Knoten (Blank Nodes) können eingeführt werden, um mehrwertige Beziehungen zu modellieren



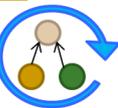
# RDF – Listen

- Allgemeine Datenstrukturen zur Aufzählung von beliebigen Ressourcen und Literalen
- Dienen lediglich einer verkürzten Schreibweise (keine zusätzliche semantische Ausdruckskraft)
- Zwei Möglichkeiten
  - *Collections*: geschlossene Listen, d.h. Hinzufügen neuer Elemente nicht möglich
  - *Container*: offene Listen, d.h. Hinzufügen neuer Elemente möglich
    - rdf:Bag: ungeordnete Elementmenge
    - rdf:Seq: geordnete Elementmenge
    - rdf:Alt: Auswahl von Einzelelementen (ein relevantes Element)



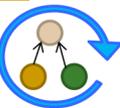
# RDF – Eigenschaften, Zusammenfassung

- Unabhängigkeit
  - Jede Property ist Ressource → eigene Properties möglich
- Austauschbarkeit
  - Basis ist XML → leichte Kommunikation möglich
- Skalierbarkeit
  - Statement aus drei Teilen → große Mengen verarbeitbar
- Properties sind Ressourcen
  - Können wiederum eigene Properties haben
- Werte können Ressourcen sein
  - Können wieder eigene Properties haben
- Statements können Ressourcen sein
  - Können wiederum auch eigene Properties haben



# RDF – Bewertung

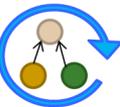
- Weitläufig unterstützter Standard (W3C Recommendation) für Datenarchivierung und Datenaustausch
  - RDF-Tools
  - RDF APIs (Jena, Redland, ...)
  - RDF Stores / Triple-Stores (Virtuoso, Sesame, Oracle...)
  - Grundlage für RSS 1.0, XMP (Adobe), ...
- Ermöglicht weitgehend syntaxunabhängige Darstellung verteilter Informationen in einem graphbasierten Datenmodell
- RDF bietet keine Möglichkeit zur Kodierung von Schemawissen ...



# RDF Schema (RDF/S)

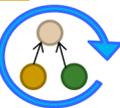
- RDF ermöglicht einfache Aussagen über Ressourcen, Properties und Werte
- Erweiterung von RDF, selbst wiederum in RDF beschrieben
- RDF Schema stellt Mechanismen zur Verfügung, um ein Vokabular zu definieren:
  - Definition von Klassen (*rdfs:Class*)
  - Anordnung von Klassen in einer Taxonomie (*rdfs:subClassOf*)
  - Zuordnung von Instanzen zu Klassen (*rdf:type*)
  - Definition von Properties (*rdfs:Property*)
  - Taxonomische Anordnung von Properties (*rdfs:subPropertyOf*)
  - Einschränkung von Werten für Properties (*rdfs:domain*, *rdfs:range*)

<http://www.w3.org/TR/rdf-schema/>



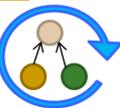
# RDF Schemadefinitionen

- Durch die Property ***rdf:type*** wird eine Instanz einer Klasse zugeordnet
  - `<MichaelHartung> <rdf:type> <PostDoc> .`
  - `<ErhardRahm> <rdf:type> <Professor> .`
- Eine Ressource wird als Klasse definiert indem man sie mithilfe von ***rdf:type*** der vordefinierten Klasse ***rdfs:Class*** zuordnet
  - `<Person> <rdf:type> <rdfs:Class> .`
- Durch ***rdfs:subClassOf*** wird einer Klasse eine Superklasse zugeordnet
  - `<Mitarbeiter> <rdfs:subClassOf> <Person> .`
  - `<PostDoc> <rdfs:subClassOf> <Mitarbeiter> .`
  - `<Professor> <rdfs:subClassOf> <Mitarbeiter> .`



# RDF Schemadefinitionen

- Um Eigenschaften zu einer Property zu definieren, ordnet man sie der Klasse ***rdf:Property*** zu
  - `<verantwortlichFür> <rdf:type> <rdf:Property> .`
  - `<hältVL> <rdf:type> <rdf:Property> .`
- Mittels ***rdfs:subPropertyOf*** können Hierarchien für Eigenschaften definiert werden
  - `<hältVL> <rdfs:subPropertyOf> <verantwortlichFür> .`
- Definitions- bzw. Wertebereich einer Property sind über ***rdfs:domain* / *rdfs:range*** möglich
  - `<verantwortlichFür> <rdfs:domain> <Mitarbeiter> .`
  - `<verantwortlichFür> <rdfs:range> <Lehrveranstaltung> .`
  - `<hältVL> <rdfs:domain> <Mitarbeiter> .`
  - `<hältVL> <rdfs:range> <Vorlesung> .`



# Vordefinierte Klassen in RDF Schema

## ■ **rdfs:Class**

- Alle Klassen sind Instanzen dieser Klasse

## ■ **rdfs:Resource**

- Die allgemeinste Klasse, alle anderen Klassen sind Subklassen dieser Klasse

## ■ **rdfs:Literal**

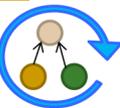
- Die Klasse aller einfachen Werte, eine Subklasse von rdfs:Resource

## ■ **rdfs:Datatype**

- Klasse von Datentypen wie Integer, Boolean usw., jede Instanz von rdfs:Datatype ist eine Subklasse von rdfs:Literal

## ■ **rdf:Property**

- Die Klasse aller Properties



# Vordefinierte Properties in RDF Schema

## ■ **rdf:type**

- Ordnet eine Instanz einer Klasse zu

## ■ **rdfs:subClassOf**

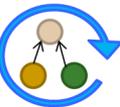
- Subklassen-Beziehung: Eine Klasse K ist Subklasse einer anderen Klassen K' genau dann, wenn alle Instanzen von K auch Instanzen von K' sind

## ■ **rdfs:range**

- Ordnet einer Property eine Klasse möglicher Werte zu (Wertebereich)

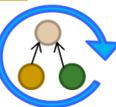
## ■ **rdfs:domain**

- Ordnet einer Property eine Klasse von Ressourcen zu, die diese Property haben können (Definitions-bereich)



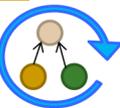
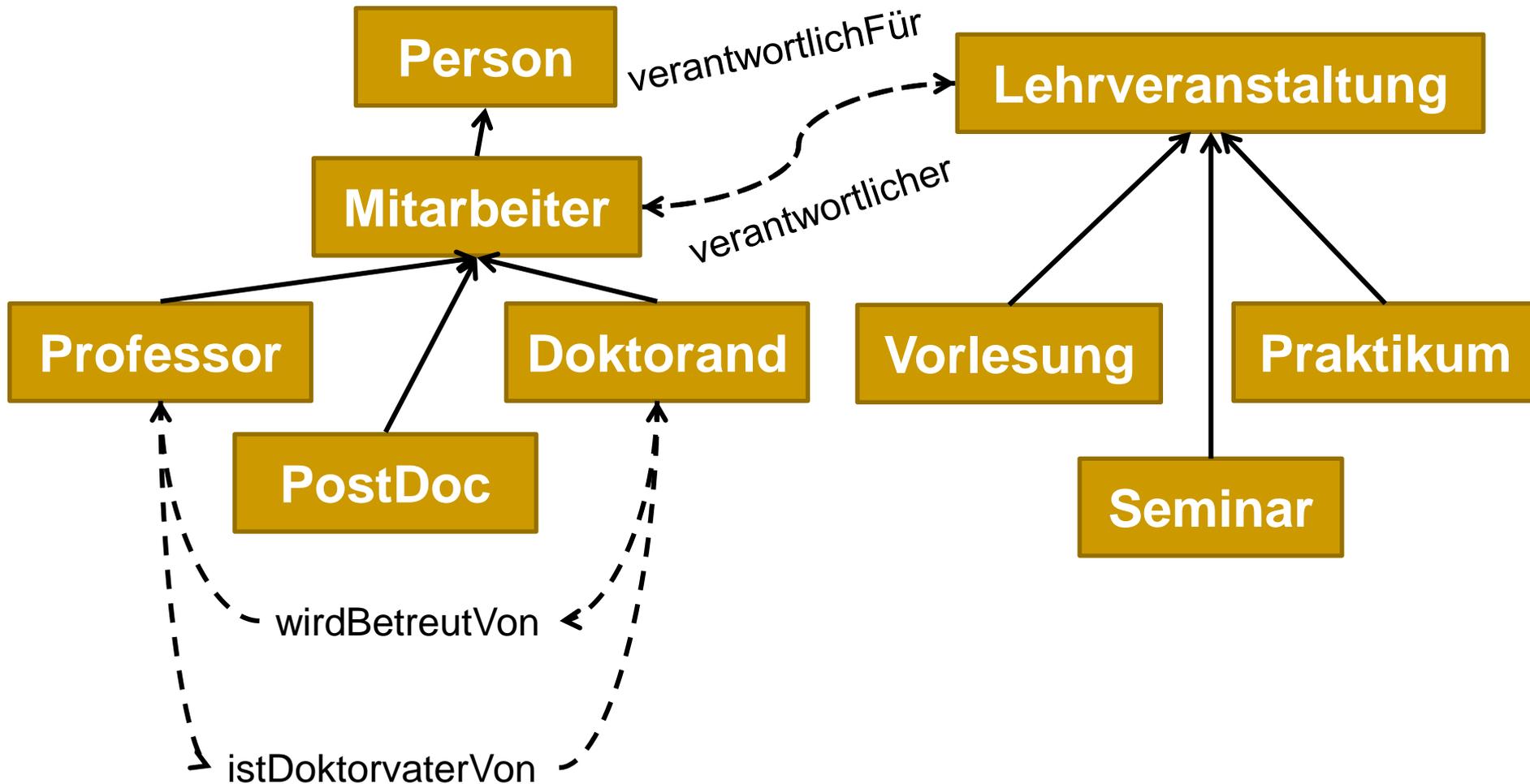
# Vordefinierte Properties in RDF Schema

- **rdfs:subPropertyOf**
  - Eine Property P ist Subproperty einer anderen Property P' genau dann, wenn alle Ressourcen, die in Eigenschaft P zueinander stehen, auch in Eigenschaft P' , zueinander stehen
- **rdfs:label**
  - für den Menschen verständlicher Bezeichner
- **rdfs:comment**



# Gesamtbeispiel – RDF Schema

- Rückgriff: „Ontologie“ für Lehrstühle



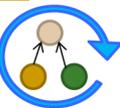
# Klassen und Klassenhierarchie

```
<Person> <rdf:type> <rdfs:Class> .  
<Mitarbeiter> <rdf:type> <rdfs:Class> .  
<Professor> <rdf:type> <rdfs:Class> .  
<PostDoc> <rdf:type> <rdfs:Class> .  
<Doktorand> <rdf:type> <rdfs:Class> .
```

```
<Lehrveranstaltung> <rdf:type> <rdfs:Class> .  
<Vorlesung> <rdf:type> <rdfs:Class> .  
<Seminar> <rdf:type> <rdfs:Class> .  
<Praktikum> <rdf:type> <rdfs:Class> .
```

```
<Mitarbeiter> <rdfs:subClassOf> <Person> .  
<Professor> <rdfs:subClassOf> <Mitarbeiter> .  
<PostDoc> <rdfs:subClassOf> <Mitarbeiter> .  
<Doktorand> <rdfs:subClassOf> <Mitarbeiter> .
```

```
<Vorlesung> <rdfs:subClassOf> <Lehrveranstaltung> .  
<Seminar> <rdfs:subClassOf> <Lehrveranstaltung> .  
<Praktikum> <rdfs:subClassOf> <Lehrveranstaltung> .
```

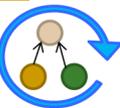


# Properties

```
<verantwortlichFür> <rdf:type> <rdf:Property> .  
<verantwortlicher> <rdf:type> <rdf:Property> .  
<istDoktorVaterVon> <rdf:type> <rdf:Property> .  
<wirdBetreutVon> <rdf:type> <rdf:Property> .
```

```
<verantwortlichFür> <rdfs:domain> <Mitarbeiter> .  
<verantwortlichFür> <rdfs:range> <Lehrveranstaltung> .  
<verantwortlicher> <rdfs:domain> <Lehrveranstaltung> .  
<verantwortlicher> <rdfs:range> <Mitarbeiter> .
```

```
<istDoktorVaterVon> <rdfs:domain> <Professor> .  
<istDoktorVaterVon> <rdfs:range> <Doktorand> .  
<wirdBetreutVon> <rdfs:domain> <Doktorand> .  
<wirdBetreutVon> <rdfs:range> <Professor> .
```



# Instanziierung

```
<Rahm> <rdf:type> <Professor> .  
<Hartung> <rdf:type> <PostDoc> .  
<Arnold> <rdf:type> <Doktorand> .  
<Groß> <rdf:type> <Doktorand> .  
<Köpcke> <rdf:type> <Doktorand> .  
<Kolb> <rdf:type> <Doktorand> .  
<Maßmann> <rdf:type> <Doktorand> .
```

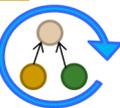
```
<DBS1> <rdf:type> <Vorlesung> .  
<IDBS1> <rdf:type> <Vorlesung> .  
<OnMa> <rdf:type> <Vorlesung> .  
<DWH-Prak> <rdf:type> <Praktikum> .  
...
```

```
<Rahm> <verantwortlichFür> <DBS1> .  
<Hartung> <verantwortlichFür> <OnMa> .  
<Groß> <wirdBetreutVon> <Rahm> .  
...
```



# RDFS Zusammenfassung

- RDF Schema spezifiziert ein Datenmodell, über das RDF-Statements entworfen werden können
- Mehr als XML:
  - (kleine) ontologische Einigung auf Modellierungsprimitive
  - Möglichkeit eigene Vokabulare zu definieren
- Nächste Schritte:
  - mehr Logik
  - Regeln, Einschränkungen und Abhängigkeitsbeziehungen



# Web Ontology Language (OWL)

- OWL ist eine Ontologie-Beschreibungssprache
  - OWL geht über RDF Schema hinaus
  - Syntax von OWL ist RDF
- Konstrukte beschrieben in <http://www.w3.org/2002/07/owl>
  - OWL 1 W3C Recommendation seit 2004
  - Neuer Standard seit 2009: OWL 1.1/ OWL 2
- Ausgewählte Konstrukte von OWL:

minCardinality

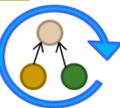
equivalentClass      unionOf      complementOf      SymmetricProperty

equivalentProperty      intersectionOf      InverseFunctionalProperty

sameIndividualAs      differentFrom      inverseOf      FunctionalProperty      allValuesFrom

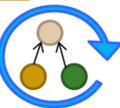
allDifferent      disjointWith      TransitiveProperty      someValuesFrom

maxCardinality



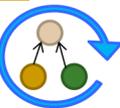
# Limitierungen von RDF Schema

- Lokale Eigenschaften
  - rdfs:range definiert Wertebereich einer Property für alle Klassen (z.B. <Tier> <isst> <Nahrung>)
  - Nicht möglich Einschränkungen zu spezifizieren, die nur auf bestimmte Klassen zutreffen
  - <Pflanzenfresser> und <Fleischfresser> als Subklassen von <Tier> sowie <Pflanze> / <Fleisch> von <Nahrung>
    - Problem: Pflanzenfresser sollten nur Pflanzen essen !!
- Kardinalitätsrestriktionen
  - Spezifikation von Kardinalitäten nicht möglich
    - eine Person hat exakt zwei Eltern
    - eine Lehrveranstaltung wird von mindestens einem Dozenten gehalten



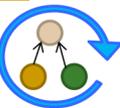
# Limitierungen von RDF Schema

- Spezielle Charakteristiken von Eigenschaften
  - Transitivität (z.B. „ist größer als“)
  - Eindeutigkeit (z.B. „ist Mutter von“)
  - Inverse (z.B. „isst“  $\leftrightarrow$  „wird gegessen“)
- Disjunktion
  - Spezifikation von Subklassen möglich jedoch ohne Angabe von Disjunktheiten
    - Pflanzenfresser / Fleischfresser als disjunkte Subklassen von Tier
- Mengenoperationen
  - Nutzung von Mengenoperationen (Union, Intersect, ...) um neue Klassen zu erzeugen
    - Pflanzenfresser sind *alle Tiere die lediglich Pflanze als Nahrung aufweisen*
    - Doktoranden / PostDocs / Professoren ergeben alle Mitarbeiter



# Drei Ausprägungen von OWL

- W3C OWL Working Group spezifizierte drei Subsprachen von OWL
- Jede Sprache zielt auf andere Aspekte / Anforderungen ab
  - OWL Full
    - Alle OWL Konstrukte erlaubt, Kombination mit RDF/S möglich
    - So ausdrucksstark, dass es unentscheidbar ist
  - OWL DL
    - Korrespondenz mit Description Logic, eingeschränkte Nutzung von Konstrukten, dadurch effizientes Reasoning möglich
    - Verlust der Kompatibilität mit RDF
  - OWL Lite
    - Weitere Restriktionen: keine Disjunktheit, beliebige Kardinalitäten
    - Leichter verständlich und umsetzbar
    - Ausdrucksstärke reduziert

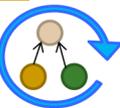


# Klassen

- Definition über **owl:Class**
  - **owl:Class** ist Subklasse von **rdfs:Class**
  - Subklassenbeziehungen wie in RDF/S
- Disjunktion mittels **owl:disjointWith**

```
<owl:Class rdf:about="#Doktorand">  
  <rdf:subClassOf rdf:resource="#Mitarbeiter"/>  
  <owl:disjointWith rdf:resource="#PostDoc"/>  
  <owl:disjointWith rdf:resource="#Professor"/>  
</owl:Class>
```

- Äquivalenz von Klassen durch **owl:equivalentClass** / **owl:sameAs**
- Zwei Spezialklassen: **owl:Thing**, **owl:Nothing**



# Properties

- **Zwei Typen von Properties**

- **Object Properties**

- Beziehungen zwischen zwei Objekten

- „verantwortlichFür“:

```
<owl:ObjectProperty rdf:ID="verantwortlichFür">  
  <rdfs:domain rdf:resource="#Mitarbeiter"/>  
  <rdfs:range rdf:resource=  
    "#Lehrveranstaltung"/>  
</owl:ObjectProperty>
```

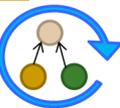
- **Datatype Properties**

- Zuordnung von einfachen Attributwerten

- „age“: <owl:DatatypeProperty rdf:ID="age">

```
  <rdfs:range  
    rdf:resource="http://www.w3.org/2001/XMLSchema  
    #nonNegativeInteger"/>
```

```
</owl:DatatypeProperty>
```



# Weitere Charakteristiken für Properties

- Inverse durch **owl:inverseOf**

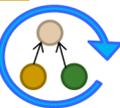
```
<owl:ObjectProperty rdf:ID="verantwortlicher">  
  <rdfs:domain rdf:resource= "#Lehrveranstaltung"/>  
  <rdfs:range rdf:resource="#Mitarbeiter"/>  
  <owl:inverseOf rdf:resource="#istVerantwortlichFür"/>  
</owl:ObjectProperty>
```

- Äquivalente Properties mit **owl:equivalentProperty**

```
<owl:ObjectProperty rdf:ID="„wirdGeleitetVon">  
  <owl:equivalentProperty rdf:resource="#verantwortlicher"/>  
</owl:ObjectProperty>
```

- Weitere

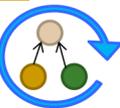
- **owl:TransitiveProperty**
- **owl:SymmetricProperty**
- ...



# Restrictions und Kardinalitäten

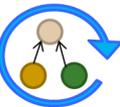
- Restrictions
  - für eine Property in Bezug auf eine Klasse
- Kardinalität
  - **owl:cardinality** für exakte Anzahl
  - **owl:minCardinality**, **owl:maxCardinality** für untere und obere Grenze

```
<owl:Class rdf:about="#Doktorand">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#wirdBetreutVon"/>
      <owl:cardinality
        rdf:datatype=" http://www.w3.org/2001/XMLSchema
          #nonNegativeInteger ">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```



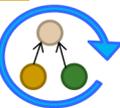
# Mengenoperationen für neue Klassen

- Vereinigung: **owl:unionOf**
  - Die Klasse *AlleMitarbeiter* ist die Vereinigung der Klassen *Professor*, *PostDoc* sowie *Doktorand*
- Durchschnitt: **owl:intersectionOf**
  - Die Klasse *Pflanzenfresser* ist der Durchschnitt der Klasse *Tier* und der Klasse aller Objekte die lediglich den Wert *Pflanzen* für die Property *isst* aufweisen
- Komplement: **owl:complementOf**
  - Selektiert alle Individuen einer Domäne, die nicht Mitglied einer bestimmten Klasse sind
- Aufzählung: **oneOf**
  - Definition einer Klasse durch Aufzählung aller Individuen, die Mitglieder der Klasse sind



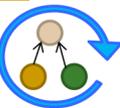
# Zusammenfassung OWL

- Vorgeschlagener Standard zur Definition von Ontologien im Semantic Web
- Baut auf RDF und RDF Schema auf
  - Syntax von RDF wird genutzt
  - Instanziierung mittels RDF Beschreibungen
  - Wiederverwendung einiger RDFS Konstrukte
- Drei Subsprachen je nach Anforderung und Applikation
- Open World Assumption
  - „Absence of information is not interpreted as negative information.“
- Keine Unique Name Assumption
  - Person1 und Person2 sind nicht notwendigerweise verschiedene Instanzen



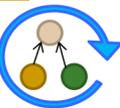
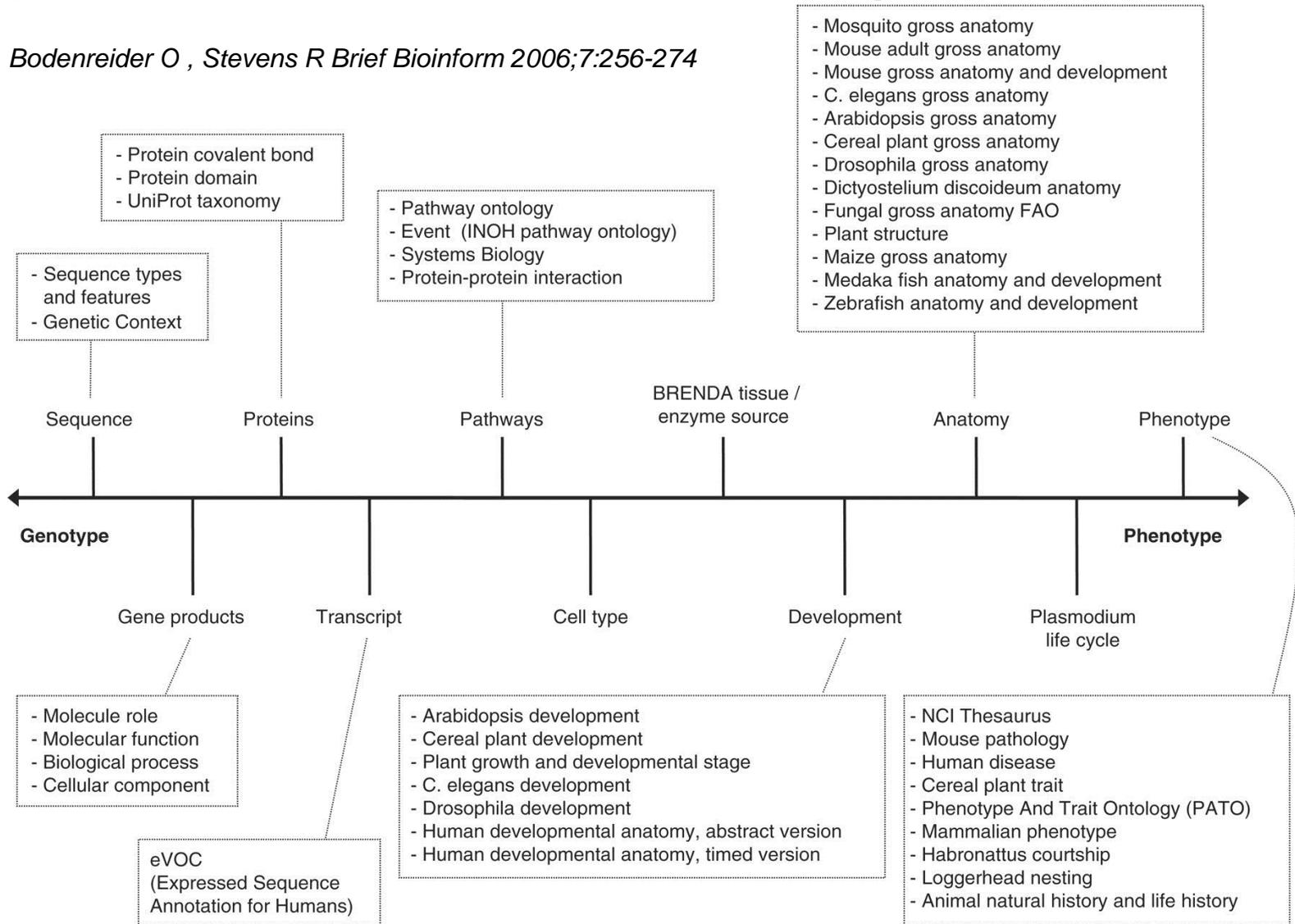
# Open Biomedical Ontologies (OBO)

- **OBO Foundry (<http://obofoundry.org/>)**
  - ❑ Infrastruktur zum Austausch/Entwicklung/Diskussion von Bioontologien
  - ❑ Ziel: keine Doppelentwicklungen (orthogonale Ontologien)
  - ❑ Prinzipien: offen, eine gemeinsame Syntax, Versionierung, kollaboratives Entwickeln, ...
- **Gemeinsame Ontologiesprache (OBO Format)**
  - ❑ Einfaches, verständliches Flatfile Format
  - ❑ Auf Anwender/Nutzer aus den Lebenswissenschaften zugeschnitten



# Spektrum von OBO Ontologien

Bodenreider O , Stevens R *Brief Bioinform* 2006;7:256-274



# OBO Format – Übersicht

## ■ OBO Flat File

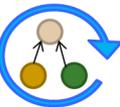
- ❑ Grundprinzip: Attribut-Wert Notation
- ❑ Jede Zeile entspricht einem Fakt

## ■ Header

- ❑ Administrative Informationen (Version, Entwickler, Erstellungsdatum, Tool, Namespace, ...)

## ■ OBO Stanzas

- ❑ *[Term]*: Zur Definition eines Terms (Klasse) samt Eigenschaften
- ❑ *[TypeDef]*: Spezifikation von Beziehungstypen, welche verwendet werden können
- ❑ *[Instances]*: Instanzen zur Ontologie



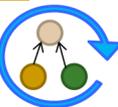
# OBO Beispiel Mouse Anatomy

```
format-version: 1.2
date: 15:07:2011 12:49
saved-by: terryh
auto-generated-by: OBO-Edit 2.1-beta13
default-namespace: adult_mouse_anatomy.gxd

[Term]
id: MA:0000001
name: mouse anatomical entity
synonym: "mouse anatomy" RELATED []

[Term]
id: MA:0000002
name: spinal cord grey matter
is_a: MA:0001112 ! grey matter
relationship: part_of MA:0000216 ! spinal
cord

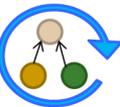
...
```



# Klassen (Terme) in OBO

## ■ Definition von Klassen/Termen

- [Term]-Stanza leitet Term ein
- Vorgegebene Menge von Attributen
  - *id*: eindeutiger Identifizierer (accession number)
  - *name*: menschenlesbarer Name
  - *def*: präzise Definition (optional), evtl. mit Quellenangabe untersetzt
  - *is\_a*: Subklassenbeziehungen zu anderen Termen
  - *relationship*: Weitere (sonstige) Beziehungen
  - *synonym*: Synonyme für den Term (mit Scope: EXACT, BROAD, NARROW, RELATED)
  - Weitere: *xref* (Querbeziehungen zu externen Quellen), *comment* (Kommentare), *is\_obsolete* (Veraltetstatus), ...



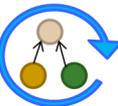
# Klassen (Terme) in OBO - Beispiel

## ■ Lehrveranstaltungen

```
[Term]
id: DB:1
name: Lehrveranstaltung
def: "Unterrichtseinheit im Rahmen des
Studiums."
```

```
[Term]
id: DB:2
name: Vorlesung
is_a: DB:1 ! Lehrveranstaltung
synonym: "VL" EXACT
```

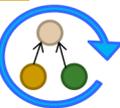
...



# Definition weiterer Beziehungstypen

- **Im Allgemeinen nur is\_a als Beziehung**
  - Wichtigster Beziehungstyp, daher vorgegeben
- **Weitere Beziehungstypen müssen angegeben (definiert) werden**
- **Stanza [Typedef]**
  - *id*: eindeutiger Identifier
  - *name*: Name der Beziehung
  - *xref*: Querverweis auf andere Ontologie
  - *is\_transitive*: Transitivitätseigenschaft

```
[Typedef]
id: wirdBetreutVon
name: wirdBetreutVon
is_transitive: false
```



# Beispiel mit mehreren Beziehungstypen

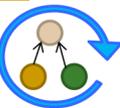
## ■ Mitarbeiter

```
[Term]
id: DB:5
name: Mitarbeiter
```

```
[Term]
id: DB:6
name: Doktorand
is_a: DB:5 ! Mitarbeiter
relationship: wirdBetreutVon DB:7 ! Professor
```

```
[Term]
id: DB:7
name: Professor
is_a: DB:5 ! Mitarbeiter
```

```
[Term]
id: DB:8
name: PostDoc
is_a: DB:5 ! Mitarbeiter
```



# Zusammenfassung

## ■ Ontologiesprachen

- ❑ Umsetzung von Ontologien
- ❑ Verschiedene Sprachen verfügbar
- ❑ Mehr oder weniger Ausdrucksstärke (Was wird unterstützt oder nicht)
- ❑ Domänenabhängig: welche Sprache wird wo akzeptiert

## ■ F-Logic

- ❑ Logik- bzw. objektorientiert

## ■ RDF/RDFS/OWL

- ❑ Semantic Web

## ■ OBO

- ❑ Lebenswissenschaften

