

# Intelligente Systeme im World Wide Web

## Web Ontology Language OWL: RDF Syntax

Folien zur Vorlesung im Sommersemester 2006

Pascal Hitzler

Institut für angewandte Informatik und Formale  
Beschreibungsverfahren (AIFB)

Universität Karlsruhe (TH)

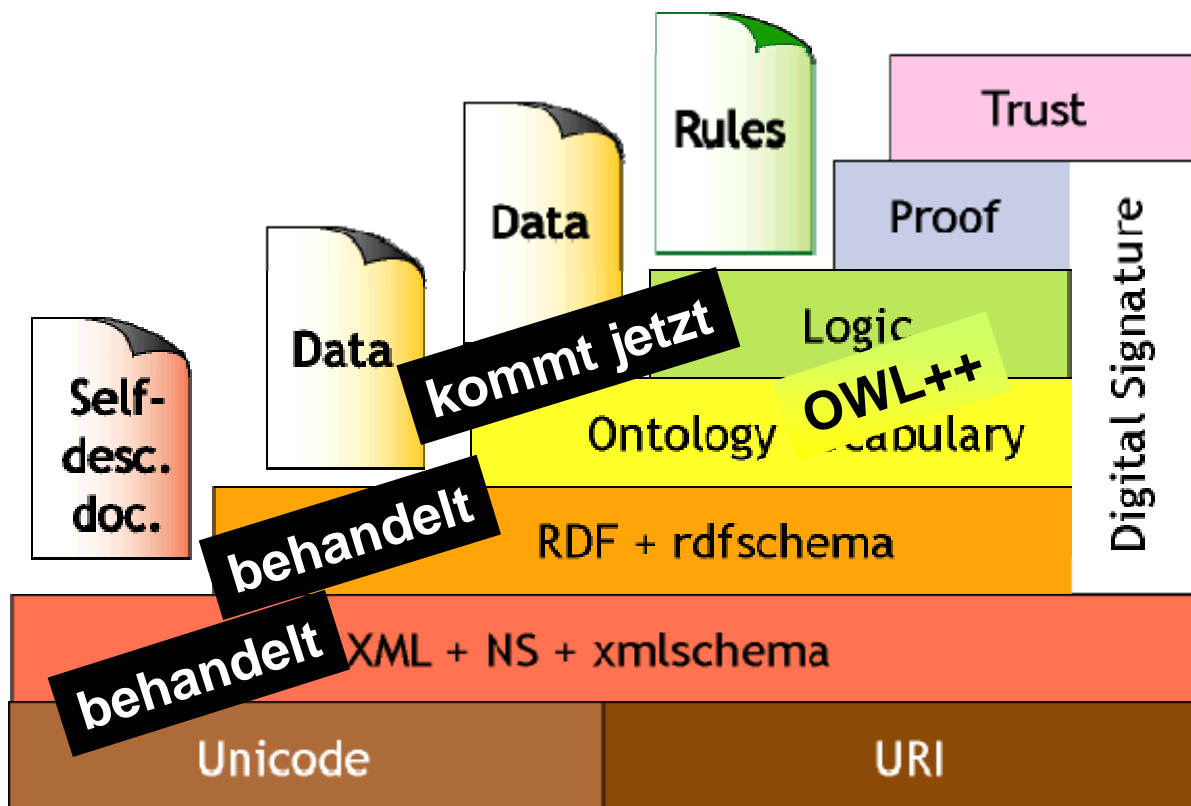
## Vorstellung

### Dr. Pascal Hitzler

- 1998 Diplom Mathematik, Tübingen
- 2001 Dissertation in Mathematik, Cork, Irland
- 2005 Habilitation in Informatik, TU Dresden  
Schwerpunkt Künstliche Intelligenz
- seit Mitte 2004 als Projektleiter am AIFB
- **Themen:**  
Wissensverarbeitung, formale Grundlagen bis zur Anwendung  
*Semantic Web*  
*Nonmonotonic Reasoning*  
*Neurosymbolische Integration*  
*Begriffsstrukturen*  
*Theorie der Semantik von Programmiersprachen*  
...
- <http://www.pascal-hitzler.de>



# Die Semantic Web Schichttorte



## Inhalt der nächsten 6 Sitzungen

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
  - c. OWL Varianten
  - d. Weitere Ressourcen
- II. Logik (Wiederholung)
  - a. Syntax
  - b. Semantik
  - c. Beweistheorie
- III. OWL – Semantische Grundlagen
  - a. Beschreibungslogiken
  - b. Beweistheorie
- IV. Ontologiesprache F-Logik

## Ontologie – philosophisch

- Ontologie einer Theorie A: notwendige Bedingungen ans Sein oder das Seiende für die Gültigkeit der Theorie A
- Ontologie ist die Lehre vom Sein qua Sein
- Fragestellungen:
  - Was ist das ‚Sein‘? Was bedeutet ‚sein‘? (Intensionale Definition)
  - Was hat ‚Sein‘? (Extensionale Definition)
  - Wie unterteilt sich das Seiende?

Aristoteles (Sokrates), Thomas von Aquin, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, ...

## Ontologie – informatisch

Gruber 93:

An Ontology is a

formal specification

of a shared

conceptualization

of a domain of interest

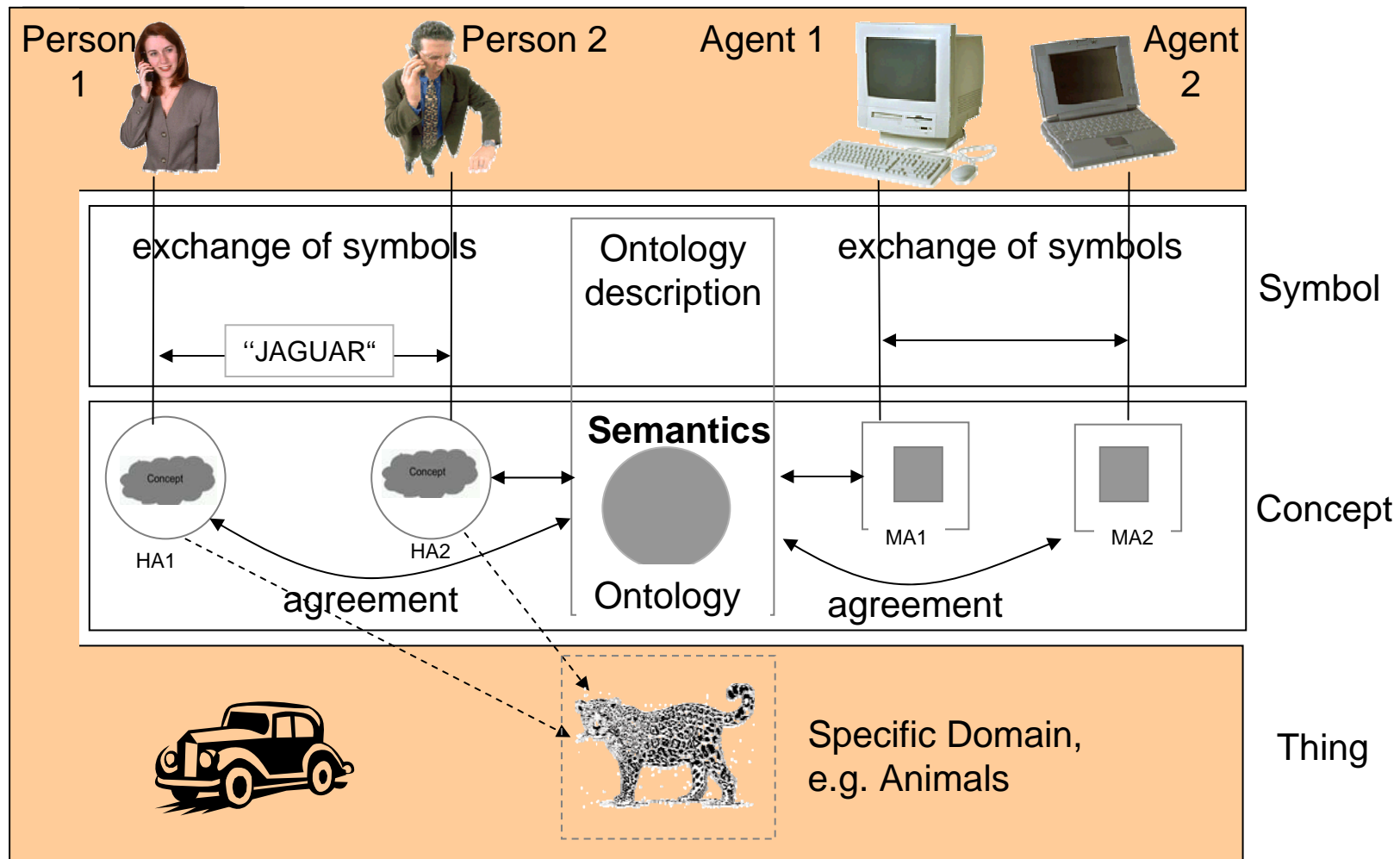
⇒ machine-understandable

⇒ group of people

⇒ about concepts

⇒ between general  
description and individual  
use

# Ontologie und Kommunikation

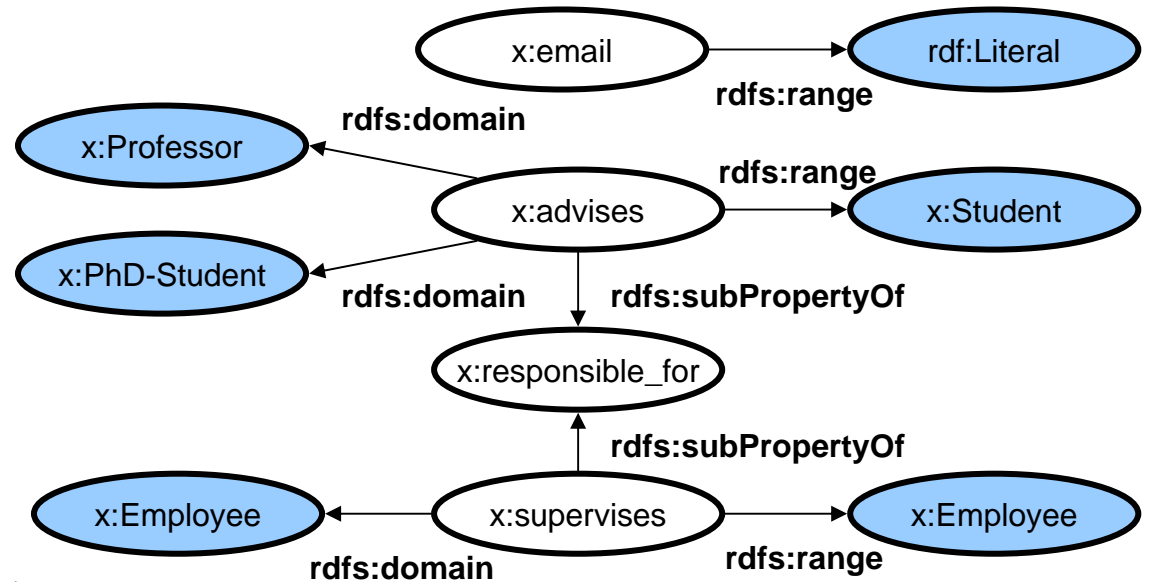


## Ontologie – praktisch. Einige Anforderungen

- Begriffshierarchie (Taxonomie): Klassen, Begriffe
- binäre Relationen zwischen Begriffen: Properties, Roles
- Vererbung: is-a etc.
  
- Datentypen (z.B. Zahlen): concrete domains
- Eigenschaften von Relationen (z.B. range, transitive)
- logische Ausdrucksmittel
  
- **Semantik!**

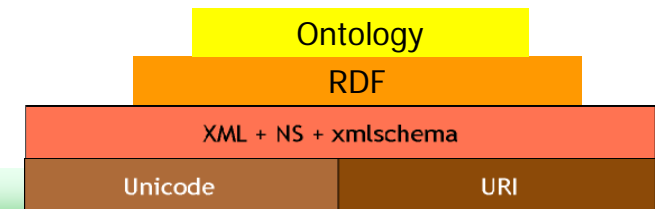
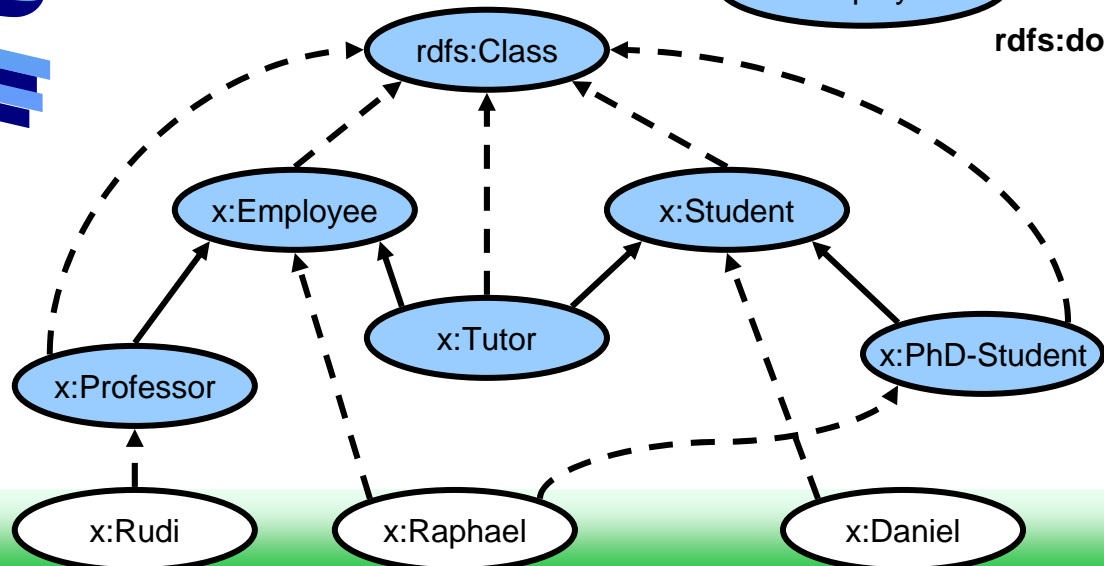
# RDF Schema – Einfache Ontologien

Declaration of properties



Declaration of classes

RDF



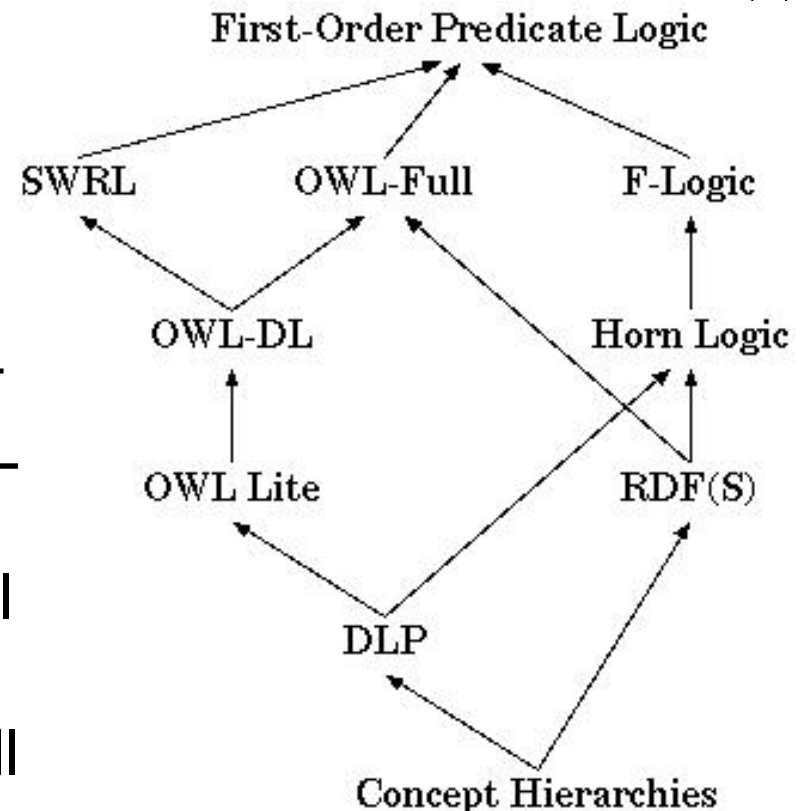
↑ subClass  
- - instantiation

## RDF Schema als Ontologiesprache

- Geeignet für einfache Ontologien
  - Für komplexere Modellierungen ungeeignet
  - → „Need for Expressivity!“
  
  - Mächtigere Sprachen:
    - OWL
    - F-Logic
    - Regelerweiterungen von OWL
- } werden wir behandeln

## OWL – Allgemeines

- W3C Recommendation seit 2004
- Semantisches Fragment von FOL
- Drei Varianten:  
 $\text{OWL Lite} \subseteq \text{OWL DL} \subseteq \text{OWL Full}$
- Keine Reifikation in OWL DL  
 RDFS ist Fragment von OWL Full
- OWL DL ist entscheidbar
- OWL DL = SHOIN(D) (Beschreibungslogik)
- W3C-Dokumente (Vorlesungswebseite) enthalten Details, die hier nicht alle angesprochen werden können.



## OWL Varianten

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einzige OWL-Teilsprache ganz RDFS
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
  - Unentscheidbar.
  - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
  - Enthält OWL Lite und ist Teilsprache von OWL Full.
  - Entscheidbar.
  - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
  - Komplexität NExpTime (worst-case).
- OWL Lite
  - Ist Teilsprache von OWL DL und OWL Full.
  - Entscheidbar.
  - Wenig ausdrucksstark.
  - Komplexität ExpTime (worst-case).

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten**
    - Kopf
    - Klassen, Rollen und Individuen
    - Klassenbeziehungen
    - komplexe Klassendefinitionen
      - **logische Konstruktoren**
      - **Rolleneinschränkungen**
    - Rolleneigenschaften
  - c. OWL Varianten
  - d. Weitere Ressourcen

## OWL Dokumente

- sind RDF Dokumente
- bestehen aus
  - Kopf mit allgemeinen Angaben
  - Rest mit der eigentlichen Ontologie

## Der Kopf eines OWL Dokumentes

- Definition von Namespaces in der Wurzel

```
<rdf:RDF
  xmlns          = "http://www.semanticweb-grundlagen.de/beispielontologie#"
  xmlns:rdf      = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd      = "http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs     = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl      = "http://www.w3.org/2002/07/owl#" >
...
</rdf:RDF>
```

## Der Kopf eines OWL Dokumentes

- Allgemeine Informationen

```
<owl:Ontology rdf:about="">
  <rdfs:comment
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    SWRC Ontologie in der Version vom Dezember 2005
  </rdfs:comment>
  <owl:versionInfo>v0.5</owl:versionInfo>
  <owl:imports rdf:resource="http://www.semanticweb-
    grundlagen.de/foo"/>
  <owl:priorVersion
    rdf:resource="http://ontoware.org/projects/swrc"/>
</owl:Ontology>
```

# Der Kopf eines OWL Dokumentes

von RDFS geerbt

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`

für Versionierung

- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`

außerdem

- `owl:imports`

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - **Klassen, Rollen und Individuen**
    - Klassenbeziehungen
    - komplexe Klassendefinitionen
      - **logische Konstruktoren**
      - **Rolleneinschränkungen**
    - Rolleneigenschaften
  - c. OWL Varianten
  - d. Weitere Ressourcen

## Klassen, Rollen und Individuen

- Die drei Bausteine von Ontologieaxiomen.
- Klassen
  - Vergleichbar mit Klassen in RDFS
- Individuen
  - Vergleichbar mit Objekten in RDFS
- Rollen
  - Vergleichbar mit Properties in RDFS

# Klassen

- Definition

```
<owl:Class rdf:ID="Professor" />
```

- vordefiniert:

- owl:Thing
    - owl:Nothing

## Individuen

- Definition durch Klassenzugehörigkeit

```
<rdf:Description rdf:ID="RudiStuder">  
<rdf:type rdf:resource="#Professor" />  
</rdf:Description>
```

- gleichbedeutend:

```
<Professor rdf:ID="RudiStuder" />
```

## abstrakte Rollen

- abstrakte Rollen werden definiert wie Klassen

```
<owl:ObjectProperty  
    rdf:ID="Zugehoerigkeit" />
```

- Domain und Range abstrakter Rollen

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">  
    <rdfs:domain rdf:resource="#Person" />  
    <rdfs:range rdf:resource="#Organisation" />  
</owl:ObjectProperty>
```

## konkrete Rollen

- konkrete Rollen haben Datentypen im Range  
`<owl:DatatypeProperty rdf:ID="Vorname" />`
- Domain und Range konkreter Rollen  
`<owl:DatatypeProperty rdf:ID="Vorname">  
 <rdfs:domain rdf:resource="#Person" />  
 <rdfs:range rdf:resource="&xsd:string" />  
</owl:DatatypeProperty>`
- Viele XML Datentypen können verwendet werden.  
Im Standard vorgeschrieben sind `integer` und `string`.

## Individuen und Rollen

```
<Person rdf:ID="RudiStuder">  
  <Zugehoerigkeit rdf:resource="#AIFB"/>  
  <Zugehoerigkeit rdf:resources="#ontoprise"/>  
  <Vorname rdf:datatype="xsd:string">Rudi</Vorname>  
</Person>
```

- Rollen sind im allgemeinen nicht funktional.

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - Klassen, Rollen und Individuen
    - **Klassenbeziehungen**
    - komplexe Klassendefinitionen
      - **logische Konstruktoren**
      - **Rolleneinschränkungen**
    - Rolleneigenschaften
  - c. OWL Varianten
  - d. Weitere Ressourcen

## Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">  
  <rdfs:subClassOf  
    rdf:resource="#Fakultaetsmitglied"/>  
</owl:Class>  
<owl:Class rdf:ID="Fakultaetsmitglied">  
  <rdfs:subClassOf rdf:resource="#Person"/>  
</owl:Class>
```

Es folgt durch Inferenz, dass Professor eine Subklasse von Person ist.

## Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf
    rdf:resource="#Fakultaetsmitglied"/>
</owl:Class>
<owl:Class rdf:ID="Buch">
  <rdfs:subClassOf rdf:resource="#Publikation"/>
</owl:Class>
<owl:Class rdf:about="#Fakultaetsmitglied">
  <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Es folgt durch Inferenz, dass Professor und Buch ebenfalls disjunkte Klassen sind.

## Einfache Klassenbeziehungen

```
<owl:Class rdf:ID="Buch">  
  <rdfs:subClassOf rdf:resource="#Publikation"/>  
</owl:Class>
```

```
<owl:Class rdf:about="#Publikation">  
  <owl:equivalentClass  
    rdf:resource="#Publication"/>  
</owl:Class>
```

Es folgt durch Inferenz, dass Buch eine Subklasse von Publication ist.

## Individuen und Klassenbeziehungen

```
<Buch rdf:ID="SemanticWebGrundlagen" >
  <Autor rdf:resource="#YorkSure" />
  <Autor rdf:resource="#PascalHitzler" />
</Buch>
<owl:Class rdf:about=="#Buch">
  <rdfs:subClassOf rdf:resource="#Publikation" />
</owl:Class>
```

Es folgt durch Inferenz, dass  
SemanticWebGrundlagen **eine** Publikation ist.

## Beziehungen zwischen Individuen

```
<Professor rdf:ID="RudiStuder" />
<rdf:Description rdf:about="#RudiStuder">
  <owl:sameAs
    rdf:resource="#ProfessorStuder" />
</rdf:Description>
```

Es folgt durch Inferenz, dass ProfessorStuder ein Professor ist.

Verschiedenheit von Individuen mittels  
**owl:differentFrom.**

## Beziehungen zwischen Individuen

```
<owl:AllDifferent>
  <owl:distinctMembers
    rdf:parseType="Collection">
    <Person rdf:about="#RudiStuder" />
    <Person rdf:about="#YorkSure" />
    <Person rdf:about="#PascalHitzler" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

Abgekürzte Schreibweise anstelle der Verwendung von mehreren `owl:differentFrom`.

Der Einsatz von `owl:AllDifferent` und `owl:distinctMembers` ist nur dafür vorgesehen.

## Abgeschlossene Klassen

```
<owl:Class rdf:about=#SekretaerinnenVonStuder>  
  <owl:oneOf rdf:parseType="Collection">  
    <Person rdf:about="#GiselaSchillinger" />  
    <Person rdf:about="#SusanneWinter" />  
  </owl:oneOf>  
</owl:Class>
```

Dies besagt, dass es nur **genau diese beiden** SekretarinnenVonStuder gibt.

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - Klassen, Rollen und Individuen
    - Klassenbeziehungen
    - **komplexe Klassendefinitionen**
      - **logische Konstruktoren**
      - **Rolleneinschränkungen**
    - Rolleneigenschaften
  - c. OWL Varianten
  - d. Weitere Ressourcen

## Logische Konstruktoren auf Klassen

- logisches und (Konjunktion):  
`owl:intersectionOf`
- logisches oder (Disjunktion):  
`owl:unionOf`
- logische negation:  
`owl:complementOf`
- Werden verwendet, um komplexe Klassen aus einfachen Klassen zu konstruieren.

## Konjunktion

```
<owl:Class rdf:about="#SekretaerinnenVonStuder">
  <owl:equivalentClass>
    <owl:intersectionOf
      rdf:parseType="Collection">
      <owl:Class rdf:about="#Sekretaerinnen"/>
      <owl:Class
        rdf:about="#AngehoeerigeAGStuder"/>
    </owl:intersectionOf>
  </owl:equivalentClass>
</owl:Class>
```

Es folgt z.B. durch Inferenz, dass alle  
SekretaerinnenVonStuder auch  
Sekretaerinnen sind.

# Disjunktion

```
<owl:Class rdf:about="#Professor">
  <owl:subClassOf>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#aktivLehrend"/>
      <owl:Class rdf:about="#imRuhestand"/>
    </owl:unionOf>
  </owl:subClassOf>
</owl:Class>
```

## Negation

```
<owl:Class rdf:about="#Fakultaetsmitglied">  
  <owl:subClassOf>  
    <owl:complementOf rdf:resource="#Publikation"/>  
  </owl:subClassOf>  
</owl:Class>
```

Sehr komplizierte Art, das Folgende auszudrücken:

```
<owl:Class rdf:about="#Fakultaetsmitglied">  
  <owl:disjointWith rdf:resource="#Publikation"/>  
</owl:Class>
```

## Rolleneinschränkungen (allValuesFrom)

- Dienen der Definition komplexer Klassen durch Rollen.

```
<owl:Class rdf:ID="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. *alle* Prüfer eine Prüfung müssen Professoren sein.

## Rolleneinschränkungen (someValuesFrom)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom rdf:resource="#Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

D.h. jede Prüfung muss *mindestens einen* Prüfer haben.

## Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        2
      </owl:maxcardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung kann *höchstens zwei* Prüfer haben.

## Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung" >
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema" />
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        3
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *mindestens drei* Themengebiete erstrecken.

## Rolleneinschränkungen (cardinalities)

```
<owl:Class rdf:about="#Pruefung" >
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatThema" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        3
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Eine Prüfung muss sich über *genau drei* Themengebiete erstrecken.

## Rolleneinschränkungen (hasValue)

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:hasValue rdf:resource="#RudiStuder"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:hasValue verweist immer auf eine konkrete Instanz.  
Dies ist äquivalent zum Beispiel auf der nächsten Folie.

## Rolleneinschränkungen (hasValue)

```
<owl:Class rdf:ID="PruefungBeiStuder">
  <rdfs:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom>
        <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about=#RudiStuder/>
        </owl:oneOf>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - Klassen, Rollen und Individuen
    - Klassenbeziehungen
    - komplexe Klassendefinitionen
      - logische Konstruktoren
      - Rolleneinschränkungen
    - **Rolleneigenschaften**
  - c. OWL Varianten
  - d. Weitere Ressourcen

## Rollenbeziehungen

```
<owl:ObjectProperty rdf:ID="hatPruefer">  
  <rdfs:subPropertyOf  
    rdf:resource="#hatAnwesenden" />  
</owl:ObjectProperty>
```

Ebenso: owl:equivalentProperty

Rollen können auch invers zueinander sein:

```
<owl:ObjectProperty rdf:ID="hatPruefer">  
  <owl:inverseOf rdf:resource="#prueferVon" />  
</owl:ObjectProperty>
```

## Rolleneigenschaften

- Domain
- Range
- Transitivität, d.h.  
 $r(a,b)$  und  $r(b,c)$  impliziert  $r(a,c)$
- Symmetrie, d.h.  
 $r(a,b)$  impliziert  $r(b,a)$
- Funktionalität  
 $r(a,b)$  und  $r(a,c)$  impliziert  $b=c$
- Inverse Funktionalität

## Domain und Range

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">  
  <rdfs:range rdf:resource="#Organisation"/>  
</owl:ObjectProperty>
```

Ist gleichbedeutend mit dem Folgenden:

```
<owl:Class rdf:about="\&owl;Thing">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#Zugehoerigkeit"/>  
      <owl:allValuesFrom rdf:resource="#Organisation"/>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```

## Domain und Range: Vorsicht!

```
<owl:ObjectProperty rdf:ID="Zugehoerigkeit">
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
<Zahl rdf:ID="Fuenf">
  <Zugehoerigkeit rdf:resource="#Primzahlen"/>
</Zahl>
```

Es folgt nun, dass `Fuenf` eine Organisation ist!

# Rolleneigenschaften



```
<owl:ObjectProperty rdf:ID="hatKollegen" >
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatProjektleiter" >
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istProjektleiterFuer" >
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
</owl:ObjectProperty>
<Person rdf:ID="YorkSure" >
  <hatKollegen rdf:resource="#PascalHitzler" />
  <hatKollegen rdf:resource="#AnupriyaAnkolekar" />
  <istProjektleiterFuer rdf:resource="#SEKT" />
</Person>
<Projekt rdf:ID="SmartWeb" >
  <hatProjektleiter rdf:resource="#PascalHitzler" />
  <hatProjektleiter rdf:resource="#HitzlerPascal" />
</Projekt>
```

## Folgerungen aus dem Beispiel

- `AnupriyaAnkolekar hatKollegen YorkSure`
- `AnupriyaAnkolekar hatKollegen PascalHitzler`
- `PascalHitzler owl:sameAs HitzlerPascal`

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - Klassen, Rollen und Individuen
    - Klassenbeziehungen
    - komplexe Klassendefinitionen
      - logische Konstruktoren
      - Rolleneinschränkungen
    - Rolleneigenschaften
  - c. OWL Varianten**
  - d. Weitere Ressourcen

## OWL Varianten

- OWL Full
  - Enthält OWL DL und OWL Lite
  - Enthält als einzige OWL-Teilsprache ganz RDFS
  - Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind.
  - Unentscheidbar.
  - Wird durch aktuelle Softwarewerkzeuge nur bedingt unterstützt.
- OWL DL
  - Enthält OWL Lite und ist Teilsprache von OWL Full.
  - Entscheidbar.
  - Wird von aktuellen Softwarewerkzeugen fast vollständig unterstützt.
  - Komplexität NExpTime (worst-case).
- OWL Lite
  - Ist Teilsprache von OWL DL und OWL Full.
  - Entscheidbar.
  - Wenig ausdrucksstark.
  - Komplexität ExpTime (worst-case).

## OWL Full

- Uneingeschränkte Nutzung aller OWL und RDFS-Sprachelemente (muss gültiges RDFS sein).
- Schwierig z.B.: nicht vorhandene Typentrennung (Klassen, Rollen, Individuen), dadurch:
  - `owl:Thing` dasselbe wie `rdfs:resource`
  - `owl:Class` dasselbe wie `rdfs:Class`
  - `owl:DatatypeProperty` Subklasse von `owl:ObjectProperty`
  - `owl:ObjectProperty` dasselbe wie `rdf:Property`

## Beispiel für Typendurchmischung in OWL Full

```
<owl:Class rdf:about="#Buch">  
  <englischerName rdf:datatype="&xsd:string">  
    book  
  </englischerName>  
  <franzoesischerName rdf:datatype="&xsd:string">  
    livre  
  </franzoesischerName>  
</owl:Class>
```

Inferenzen über solche Konstrukte werden oft nicht wirklich benötigt.

## OWL DL

- Nur Verwendung von explizit erlaubten RDFS Sprachelementen (z.B. die in unseren Beispielen).  
Nicht erlaubt: `rdfs:Class`, `rdfs:Property`
- Typentrennung. Klassen und Rollen müssen explizit deklariert werden.
- Konkrete Rollen dürfen nicht als Transitiv, Symmetrisch, Invers oder Invers Funktional deklariert werden.
- Zahlenrestriktionen dürfen nicht mit transitiven Rollen, deren Subrollen, oder inversen davon verwendet werden.

## OWL Lite

- Alle Einschränkungen für OWL DL gelten.
- Nicht erlaubt: `oneOf`, `unionOf`, `complementOf`, `hasValue`, `disjointWith`
- Zahlenrestriktionen nur mit 0 und 1 erlaubt.
- Einige Einschränkungen zum Auftreten von anonymen (komplexen) Klassen, z.B. nur im Subjekt von `rdfs:subClassOf`.

## Anfragen an OWL (nur Klassen und Rollen)

- Klassenäquivalenz
- Subklassenbeziehung
- Disjunktheit von Klassen
- globale Konsistenz (Erfüllbarkeit, Widerspruchsfreiheit)
- Klassenkonsistenz: Eine Klasse ist *inkonsistent*, wenn sie äquivalent zu `owl:Nothing` ist.
  - Dies deutet oft auf einen Modellierungsfehler hin.

```
<owl:Class rdf:about="#Buch">  
  <owl:subClassOf rdf:resource="#Publikation"/>  
  <owl:disjointWith rdf:resource="#Publikation"/>  
</owl:Class>
```

## Anfragen an OWL (mit Individuen)

- Suche nach allen Individuen, die in einer Klasse enthalten sind.
- Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?

# Inhalt

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
    - Kopf
    - Klassen, Rollen und Individuen
    - Klassenbeziehungen
    - komplexe Klassendefinitionen
      - logische Konstruktoren
      - Rolleneinschränkungen
    - Rolleneigenschaften
  - c. OWL Varianten
  - d. Weitere Ressourcen**

## OWL Werkzeuge

- Editoren
  - Protegé, <http://protege.stanford.edu>
  - SWOOP, <http://www.mindswap.org/2004/SWOOP/>
  - OWL Tools, <http://owltools.ontoware.org/>
- Inferenzmaschinen
  - Pellet, <http://www.mindswap.org/2003/pellet/index.shtml>
  - KAON2, <http://kaon2.semanticweb.org>
  - FACT++, <http://owl.man.ac.uk/factplusplus/>
  - Racer, <http://www.racer-systems.com/>
  - Cerebra, <http://www.cerebra.com/index.html>

# OWL Sprachelemente

## Kopf

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`
- `owl:imports`

## Beziehungen zwischen Individuen

- `owl:sameAs`
- `owl:differentFrom`
- `owl:AllDifferent`  
(zusammen mit `owl:distinctMembers`)

## Vorgeschriebene Datentypen

- `xsd:string`
- `xsd:integer`

# OWL Sprachelemente

## Klassenkonstruktoren und -beziehungen

- `owl:Class`
- `owl:Thing`
- `owl:Nothing`
- `rdfs:subClassOf`
- `owl:disjointWith`
- `owl:equivalentClass`
- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`

## Rollenrestriktionen

- `owl:allValuesFrom`
- `owl:someValuesFrom`
- `owl:hasValue`
- `owl:cardinality`
- `owl:minCardinality`
- `owl:maxCardinality`
- `owl:oneOf`

## OWL Sprachelemente

### Rollenkonstruktoren, -beziehungen und -eigenschaften

- `owl:ObjectProperty`
- `owl:DatatypeProperty`
- `rdfs:subPropertyOf`
- `owl:equivalentProperty`
- `owl:inverseOf`
- `rdfs:domain`
- `rdfs:range`
- `rdf:resource="&owl;TransitiveProperty"`
- `rdf:resource="&owl;SymmetricProperty"`
- `rdf:resource="&owl;FunctionalProperty"`
- `rdf:resource="&owl;InverseFunctionalProperty"`

## Weiterführende Literatur

- <http://www.w3.org/2004/OWL/>  
zentrale W3C Webseite für OWL.
- <http://www.w3.org/TR/owl-features/>  
Überblick über OWL.
- <http://www.w3.org/TR/owl-ref/>  
vollständige Beschreibung der OWL-Sprachkomponenten.
- <http://www.w3.org/TR/owl-guide/>  
zeigt, wie OWL zur Wissensmodellierung verwendet werden kann.
- <http://www.w3.org/TR/owl-semantics/>  
beschreibt die Semantik von OWL, die wir auf andere Weise später behandeln werden. Es beschreibt außerdem die abstrakte Syntax für OWL DL, die wir hier später noch ansprechen.
- Deutsche Übersetzungen mancher W3C Dokumente findet man unter  
<http://www.w3.org/2005/11/Translations/Lists/ListLang-de.html>

## Inhalt der nächsten 5 Sitzungen

- I. OWL – Syntax und allgemeines Verständnis
  - a. Was ist „Ontologie“?
  - b. Bausteine von OWL Dokumenten
  - c. OWL Varianten
  - d. Weitere Ressourcen
- II. Logik (Wiederholung)
  - a. Syntax
  - b. Semantik
  - c. Beweistheorie
- III. OWL – Semantische Grundlagen
  - a. Beschreibungslogiken
  - b. Beweistheorie
- IV. Ontologiesprache F-Logik