- ZIEL 1. VL erreicht

DL-Metrik

Ontology metrics:

Metrics

Axiom	82
Logical axiom count	49
Class count	27
Object property count	6
Data property count	0
Individual count	0
DL expressivity	SHIF

Nächste Folie

Class axioms

SubClassOf axioms count	25
EquivalentClasses axioms count	0
DisjointClasses axioms count	8
GCI count	0
Hidden GCI Count	0

Nächste Folie

Object property axioms

SubObjectPropertyOf axioms count	4
EquivalentObjectProperties axioms count	0
InverseObjectProperties axioms count	3
DisjointObjectProperties axioms count	0
FunctionalObjectProperty axioms count	1
InverseFunctionalObjectProperty axioms count	0
TransitiveObjectProperty axioms count	2
SymmetricObjectProperty axioms count	0
AsymmetricObjectProperty axioms count	0
ReflexiveObjectProperty axioms count	0
IrrefexiveObjectProperty axioms count	0
ObjectPropertyDomain axioms count	3
ObjectPropertyRange axioms count	3
SubPropertyChainOf axioms count	0

Data property axioms

SubDataPropertyOf axioms count	0

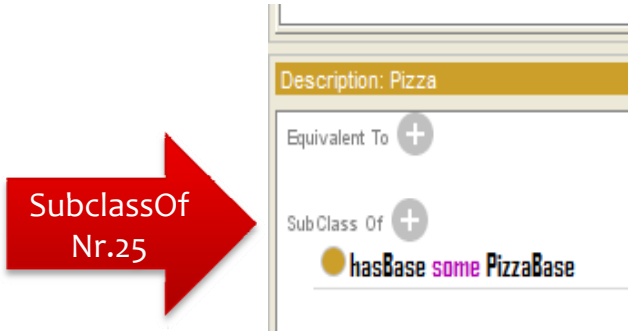
Welche Ontologie?

49 logische Axiome

Taxonomie in der Pizza-Welt

27 Klassen

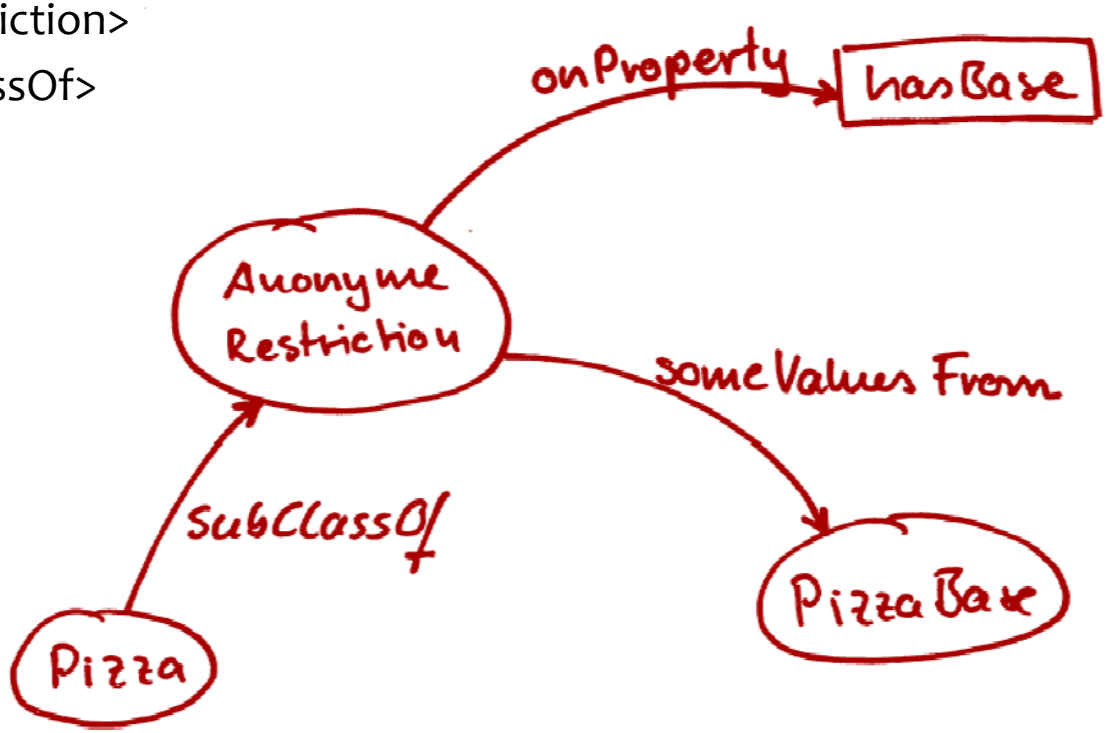
Taxonomie in der Pizza-Welt



24 SubclassOf-Beziehungen (zu Thing zählt hier nicht mit)
Welche SubclassOf-Beziehung fehlt noch?

$Pizza \sqsubseteq \exists hasBase.PizzaBase$

```
<owl:Class rdf:about="http://www.pizza.com/ontologies/pizza.owl#Pizza">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty  rdf:resource="http://www.pizza.com/ontologies/pizza.owl#hasBase"/>
        <owl:someValuesFrom rdf:resource="http://www.pizza.com/ontologies/pizza.owl#PizzaBase"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```



ObjectProperties

Ontology metrics:

Metrics

Axiom	82
Logical axiom count	49
Class count	27
Object property count	6
Data property count	0
Individual count	0
DL expressivity	SHIF

Class axioms

SubClassOf axioms count	25
EquivalentClasses axioms count	0
DisjointClasses axioms count	8
GCI count	0
Hidden GCI Count	0

Object property axioms

SubObjectPropertyOf axioms count	4
EquivalentObjectProperties axioms count	0
InverseObjectProperties axioms count	3
DisjointObjectProperties axioms count	0
FunctionalObjectProperty axioms count	1
InverseFunctionalObjectProperty axioms count	0
TransitiveObjectProperty axioms count	2
SymmetricObjectProperty axioms count	0
AsymmetricObjectProperty axioms count	0
ReflexiveObjectProperty axioms count	0
IrrefexiveObjectProperty axioms count	0
ObjectPropertyDomain axioms count	3
ObjectPropertyRange axioms count	3
SubPropertyChainOf axioms count	0

Data property axioms

SubDataPropertyOf axioms count	0

Welche?

Object property hierarchy: topObject

- topObjectProperty
  - hasIngredient
    - hasBase
    - hasTopping
  - isIngredientOf
    - isBaseOf
    - isToppingOf

Welche?

Welche?

hasBase

hasIngredient, isIngredientOf

hasBase, hasToppingOf, hasBaseOf

hasBase, hasToppingOf, hasBaseOf

4

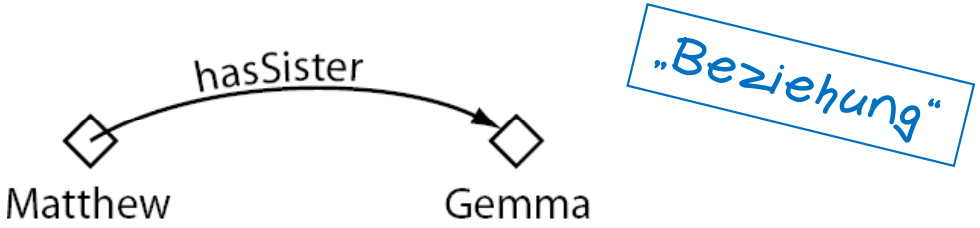
OWL-DL: Axiome über Rollen

ObjectProperty und DatatypeProperty

PR

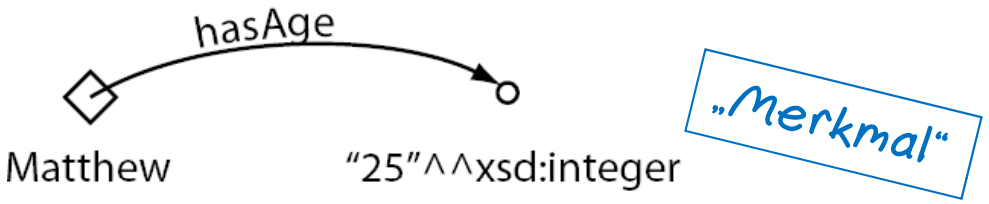
- In OWL gibt es zwei Arten von Rollen:

Abstrakte Rollen : Klasse -> Klasse (ObjectProperty)



An object property linking the individual Matthew to the individual Gemma

Konkrete Rollen: Klasse -> Datentyp (DatatypeProperty)



A datatype property linking the individual Matthew to the data literal '25', which has a type of an xsd:integer.

4

OWL-DL: Axiome über Rollen

Abstrakte Rollen „ObjectProperty“

- **Abstrakte Rollen** sind Rollen zwischen Klassen
- Beispiel: abstrakte Rolle ‚Mitgliedschaft‘

```
<owl:ObjectProperty rdf:ID=„Mitgliedschaft"/>
```

- Domain und Range abstrakter Rollen

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

Mitgliedschaft besteht zwischen einer Person und einer Organisation.

4

OWL-DL: Axiome über Rollen

Konkrete Rollen „DatatypeProperty“

- **Konkrete Rollen** haben als Range einen vordefinierten Datentypen
- Beispiel: konkrete Rolle ‚Vorname‘

```
<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, bspw. integer und string

Der Vorname einer Person ist eine Zeichenkette.

Domain und Range sind Axiome

- Domain und Range nicht als Einschränkungen sehen, die geprüft werden – sondern als Axiome über Rollen

Beispiel:

Die Rolle hasTopping habe die Domain Pizza. IceCream hat mindestens eine Relation hasTopping.

TBox

- Pizza ist eine Klasse
- IceCream ist eine Klasse
- hasTopping ist eine Property
- Domain(hasTopping) ist Pizza
- IceCream istUnterklasse von Exist hasTopping.Thing

Frage: Ist die TBox konsistent?

- Ja

Was können Sie schlussfolgern?

- IceCream ist Unterklasse von Pizza

Domain und Range sind Axiome – Beispiel in Protege

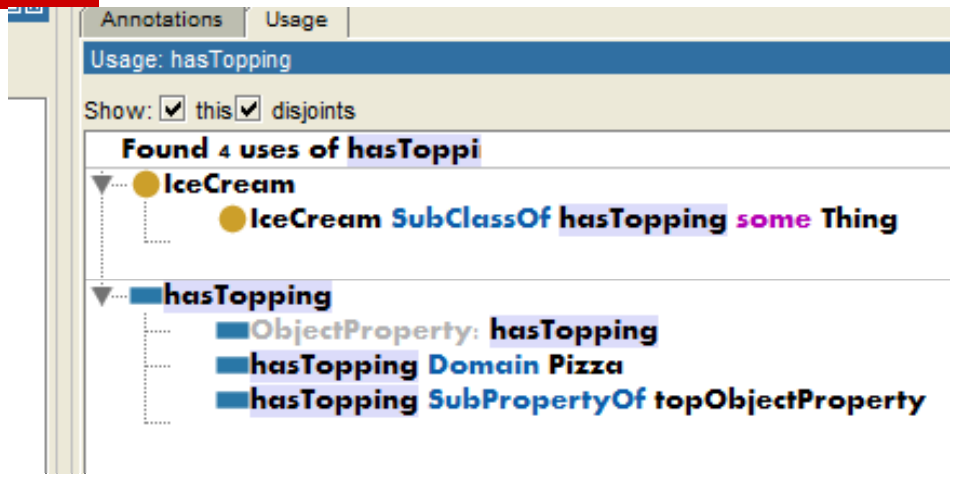
```
<owl:ObjectProperty rdf:about="&owl;topObjectProperty"/>

<owl:ObjectProperty rdf:about=„hasTopping">
    <rdfs:domain rdf:resource=„Pizza"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>

<owl:Class rdf:about="&owl;Thing"/>

<owl:Class rdf:about=„Pizza"/>

<owl:Class rdf:about=„IceCream">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource=„hasTopping"/>
        <owl:someValuesFrom rdf:resource="&owl;Thing"/>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```



Inferenz

4 OWL-DL: Axiome über Rollen

Weitere Aussagen über Rollen ObjectProperties

- ObjectProperties (Rollen zwischen Klassen) können markiert werden als :
  - transitiv,
  - symmetrisch,
  - funktional und
  - inverse funktional.

Eigenschaften von Relationen (1. Sem.?)

->Übung

  - **Beispiel**: Die abstrakte Rolle ‚grenztAn' ist symmetrisch

- **Einschränkung**: Damit die OWL DL entscheidbar bleibt, können diese Eigenschaften nicht beliebig kombiniert werden.
- Es muss gelten: Properties, die als ‚**transitiv**' markiert sind, können nicht in ihrer Cardinalität eingeschränkt werden, d.h. kein 'Functional' und keine 'min-' oder 'max-Cardinality'.
- Dies gilt ebenso für ihre Superproperties und ihre Inversen.

4

OWL-DL: Axiome über Rollen

Beispiel symmetrische Relation R, aRb → bRa

```
<owl:ObjectProperty rdf:ID=„grenztAn">
    <rdf:type rdf:resource="&owl;SymmetricProperty" />
    <rdfs:domain rdf:resource="#Region" />
    <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID=„LandkreisTeltowFläming">
    <locatedIn rdf:resource="#BundeslandBrandenburg" />
    <grenztAn rdf:resource="#LandkreisElbeElster " />
</Region>
```

- Es folgt durch Inferenz, dass auch [redacted] grenzt.

Sprachkonstrukte in OWL

- **Konstruktoren zur Beschreibung** komplexer Konzepte und Rollen aus einfachen Konzepten und Rollen
  - Konzeptkonstruktoren (Descriptions)
  - Rollenkonstruktoren
- **Axiome zum Ausdruck von** Fakten über Konzepte, Rollen und Individuen
  - Über Konzepte
    - **C ist Unterklasse von <Konzeptkonstruktor>**
    - **C und D sind äquivalent**
    - C und D haben keine gemeinsamen Elemente
  - Über Rollen
    - R hat Domain und Range
    - R ist eine Funktion, ist transitiv
  - Über Individuen
    - a ist ein C, a ist verschieden von b, Rollenzuweisung R(b,c)

5

5

Assertions in OWL-DL (Fakten)

- ABox

Abstrakte Syntax in OWL	**DL Syntax**
Individual(o type(C_1)... type(C_n))	$o \in C_i$ oder $C_i(o)$
Individual(o value(R_1 o_1)...value(R_n o_n))	$(o, o_i) \in R_i$ oder $R_i(o, o_i)$
SameIndividual($o_1 \ldots o_n$)	$\{o_1\} \equiv \cdots \equiv \{o_n\}$
DifferentIndividuals($o_1 \ldots o_n$)	$\{o_i\} \sqsubseteq \neg\{o_j\}, i \neq j$

Individuen

PR

Konzeptzuweisung C(a)

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

- gleichbedeutend:

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

$$Professor(uhlig)$$

Individuen und Rollen

PR

Rollenzuweisung R(b,c)

```
<Person rdf:ID=„DietmarUhlig">
    <istMitgliedVon  rdf:resource="#FBIuM"/>
    <hatVorname rdf:datatype="&xsd;string">Dietmar</hatVorname>
</Person>
```

$$istMitgliedVon(uhlig, FBIuM)$$

$$hatVorname(uhlig, Dietmar)$$

5 Owl:sameas

- Individuen sind identisch

```
<rdf:Description rdf:about="#William_Jefferson_Clinton">
    <owl:sameAs rdf:resource="#BillClinton"/>
</rdf:Description>
```

$$WilliamJeffersonClinton \equiv BillClinton$$

Das $(\mathcal{D})$ in $\mathcal{SHOIN}(\mathcal{D})$

- OWL unterstützt **XML Schema** primitive Datentypen, bspw. integer, real, string
- Strenge Trennung zwischen
  - ‚Objekt'-Klassen und
  - Datentypen
- Disjunkte Interpretationsdomänen
- Disjunkte ‚Objekt'Properties (Abstrakte Rollen) und Datentyp-Properties (Konrete Rollen)

Das $(\mathcal{D})$ in $\mathcal{SHOIN}(\mathcal{D})$

- OWL unterstützt **XML Schema** primitive Datentypen, bspw. integer, real, string
- Strenge Trennung zwischen
  - ‚Objekt'-Klassen und
  - Datentypen
- Disjunkte Interpretationsdomänen
- Disjunkte ‚Objekt'Properties (Abstrakte Rollen) und Datentyp-Properties (Konrete Rollen)

OWL 2

- W3C-Standard seit Oktober 2009
- **OWL 2 ist entscheidbar**
  - OWL: SHOIN(D)
  - OWL2: SROIQ(D)

Wozu

PR

Terminologien in Form des Semantic Web bieten als ‚Mehrwert':

- **Vernetzung** von Wissen mit Hilfe eindeutiger Begriffe (URI, gemeinsame OWL-Ontologie als Vokabular)

- **Inferenz** in semantischen Netzen

Next

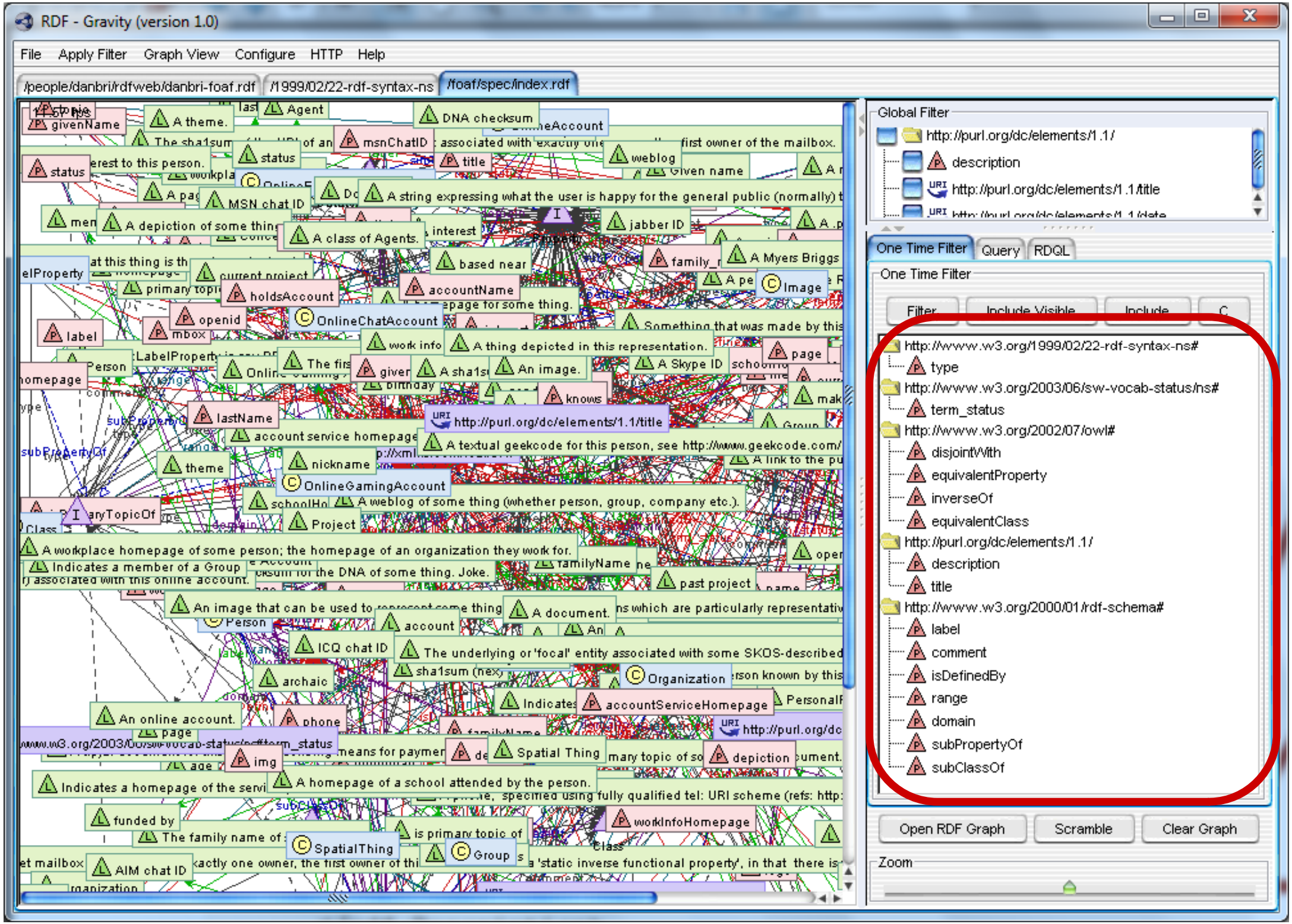
- Ontologie und die beiden Visionen des ‚Semantic Web'
- RDF, RDFS
- Web Ontology Language OWL
  - OWL lesen – Sprachkonstrukte
  - Beispiel einer Ontologie: FOAF
- Von OWL zu OWL2

Beispiel einer Ontologie: FOAF

The Friend of a Friend (FOAF) project

- The *Friend of a Friend* (FOAF) project is creating a Web of machine-readable pages describing people, the links between them and the things they create and do

- http://xmlns.com/foaf/spec/index.rdf
  - Klassen
  - (Hilfs)Instanzen
  - Properties
  - Properties zwischen Klassen
  - Properties zwischen Properties

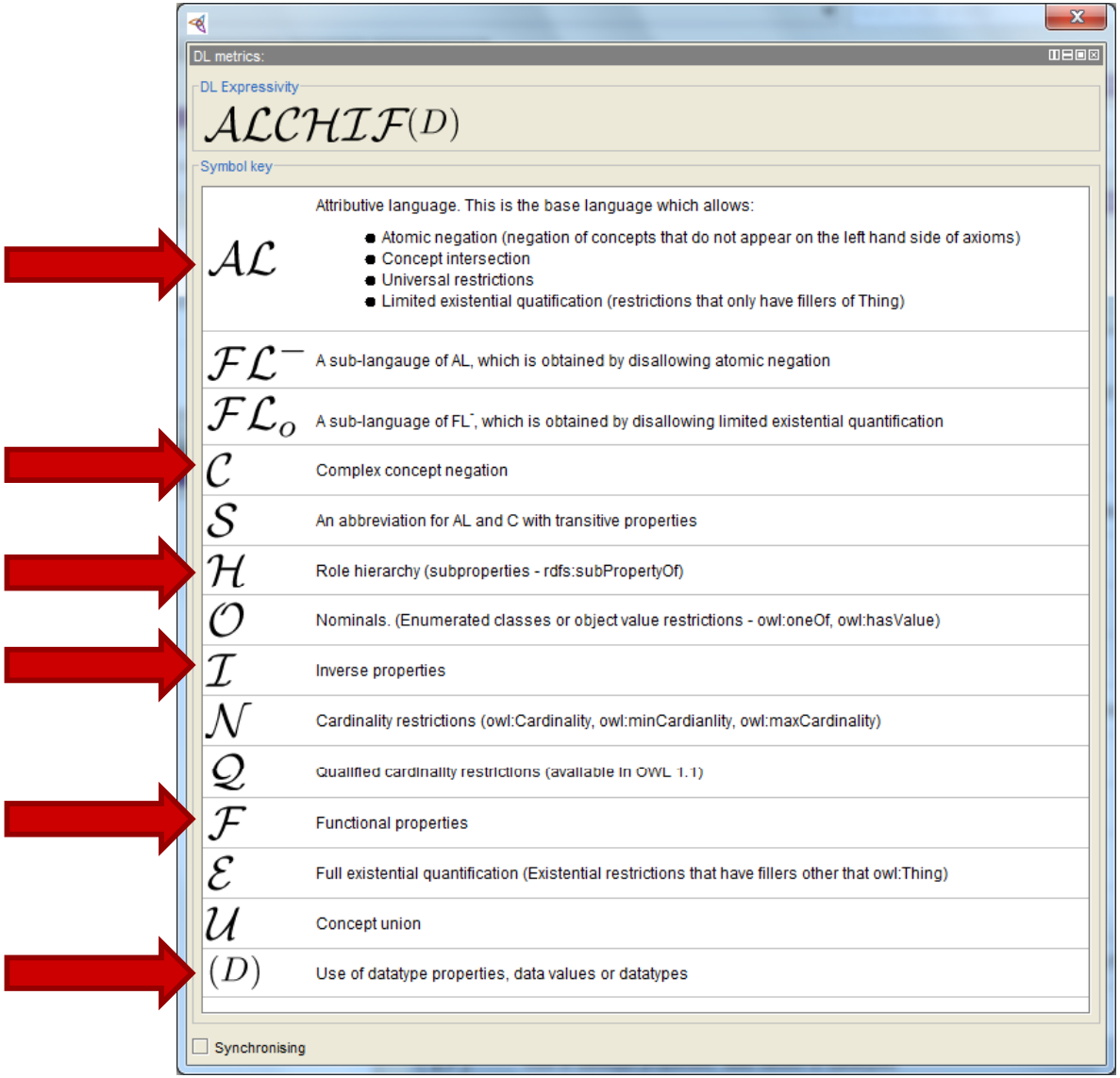

RDF - Gravity (version 1.0)
File
Apply Filter
Graph View
Configure
HTTP
Help
/people/danbri/rdfweb/danbri-foaf.rdf
/1999/02/22-rdf-syntax-ns
/foaf/spec/index.rdf
Global Filter
http://purl.org/dc/elements/1.1/
description
http://purl.org/dc/elements/1.1/title
One Time Filter
Query
RDQL
One Time Filter
Filter
Include Visible
Include
http://www.w3.org/1999/02/22-rdf-syntax-ns#
type
http://www.w3.org/2003/06/sw-vocab-status/ns#
term_status
http://www.w3.org/2002/07/owl#
disjointWith
equivalentProperty
inverseOf
equivalentClass
http://purl.org/dc/elements/1.1/
description
title
http://www.w3.org/2000/01/rdf-schema#
label
comment
isDefinedBy
range
domain
subPropertyOf
subClassOf
Open RDF Graph
Scramble
Clear Graph
Zoom

Beispiel FOAF: DL-Metrik

Metrics

Axiom	550
Logical axiom count	157
Class count	19
Object property count	40
Data property count	27
Individual count	0
DL expressivity	ALCHIF(D)

Ausdrucksstärke der in FOAF verwendeten Axiome



Beispiel FOAF: DL-Metrik

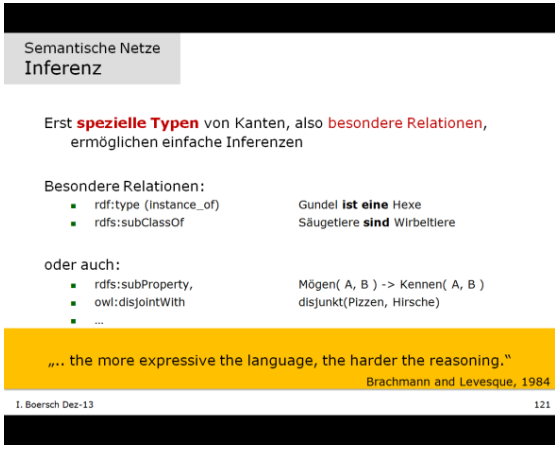
Metrics

Axiom	550
Logical axiom count	157
Class count	19
Object property count	40
Data property count	27
Individual count	0
DL expressivity	ALCHIF(D)

AL = Attributive Language (Negation, Intersection. Universal restrictions, Existenz

Beispiel FOAF: 157 Beschreibungslogische Axiome

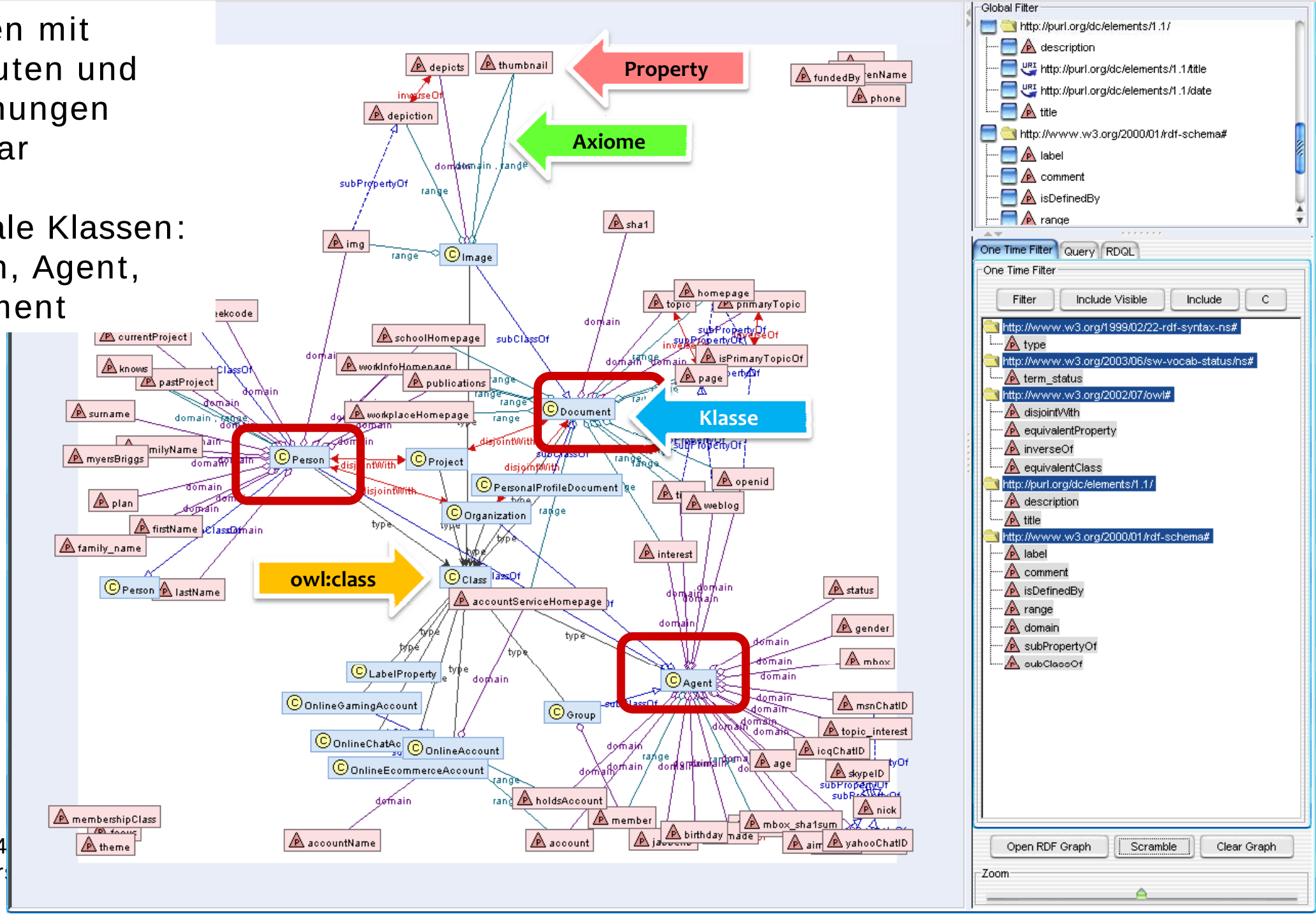
Klassen	SubClassOf axioms count	11
	EquivalentClasses axioms count	1
	DisjointClasses axioms count	4
	GCI count	0
	Hidden GCI Count	0
Abstrakte Rollen	SubObjectPropertyOf axioms count	7
	EquivalentObjectProperties axioms count	0
	InverseObjectProperties axioms count	4
	DisjointObjectProperties axioms count	0
	FunctionalObjectProperty axioms count	1
	InverseFunctionalObjectProperty axioms count	12
	TransitiveObjectProperty axioms count	0
	SymmetricObjectProperty axioms count	0
	AsymmetricObjectProperty axioms count	0
	ReflexiveObjectProperty axioms count	0
	IrrefexiveObjectProperty axioms count	0
	ObjectPropertyDomain axioms count	39
	ObjectPropertyRange axioms count	33
	SubPropertyChainOf axioms count	0
Konkrete Rollen	SubDataPropertyOf axioms count	5
	EquivalentDataProperties axioms count	0
	DisjointDataProperties axioms count	0
	FunctionalDataProperty axioms count	3
	DataPropertyDomain axioms count	15
	DataPropertyRange axioms count	22



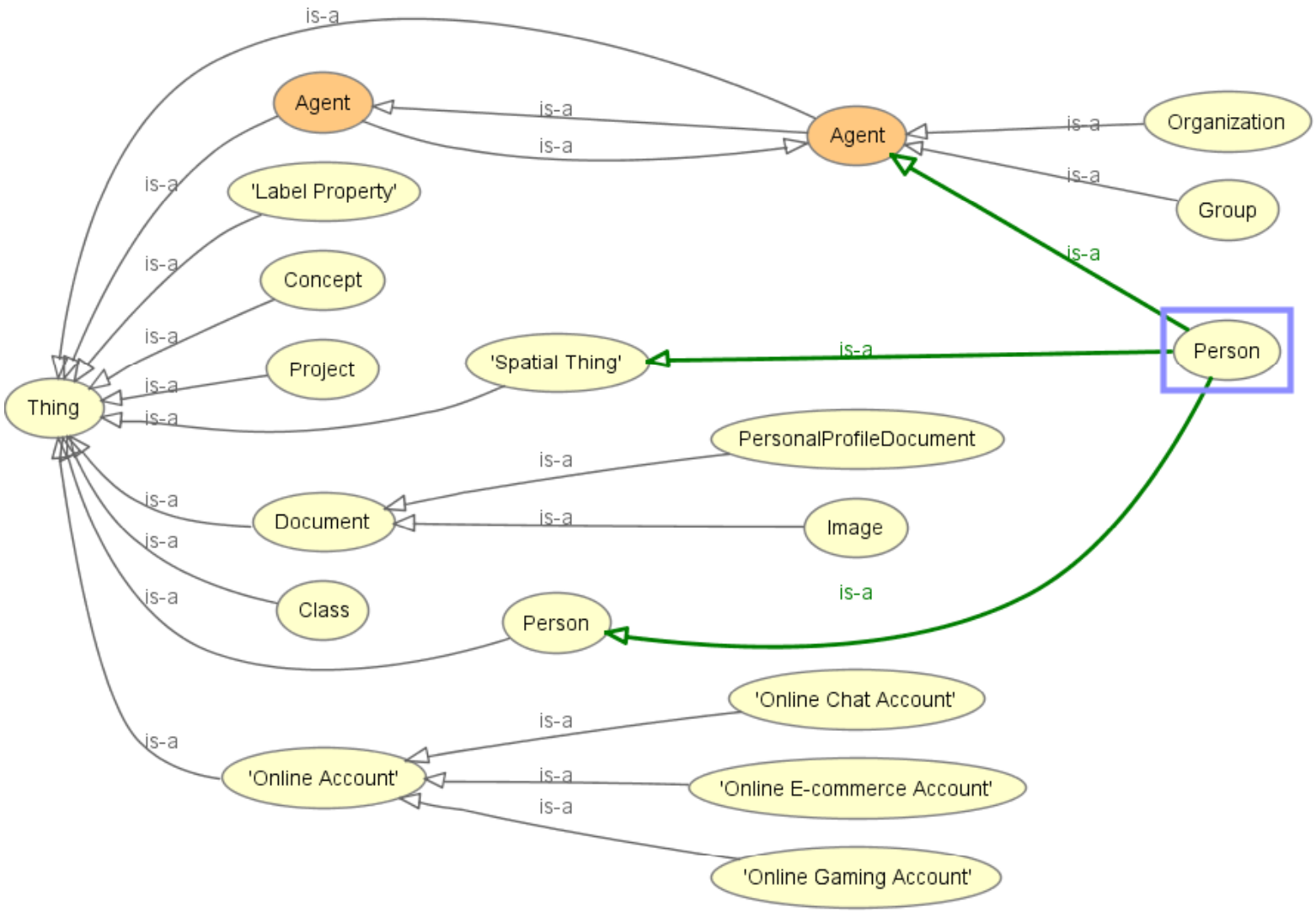
FOAF: ohne Literale, ohne URI, ohne Instanzen

Klassen mit Attributen und Beziehungen sichtbar

Zentrale Klassen: Person, Agent, Document



Klassenstruktur mit OWLViz (Protege)

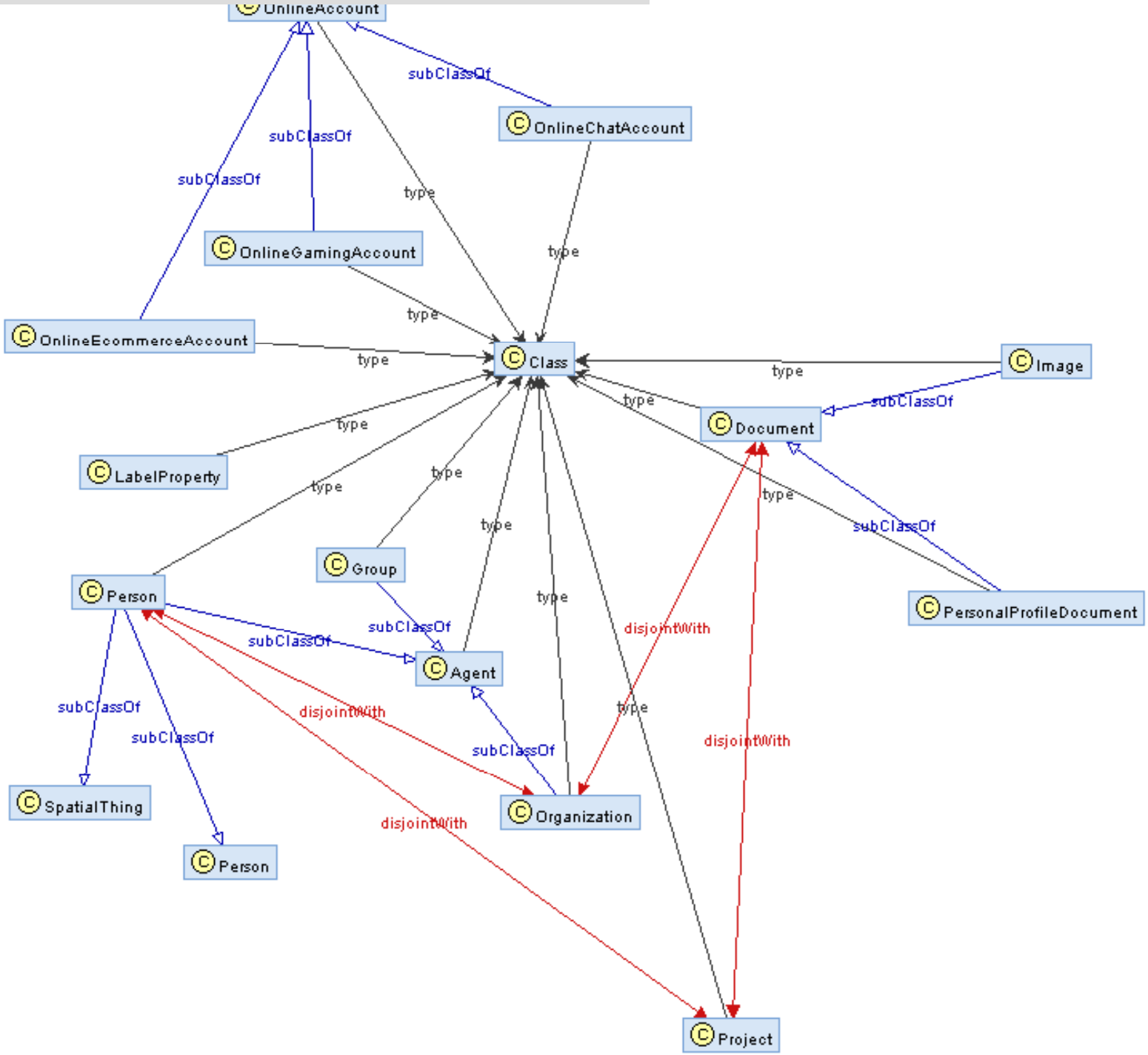


FOAF: nur Konzepte und ihre Beziehungen

Quelle der Properties:

- type aus RDF,
- subclassOf aus RDFS,
- disjointWith aus OWL

- Eine sehr kleine Polyhierarchie



FOAF: nur Properties und Instanzen: type (RDF)

- rdf:type Klassifikation der Properties

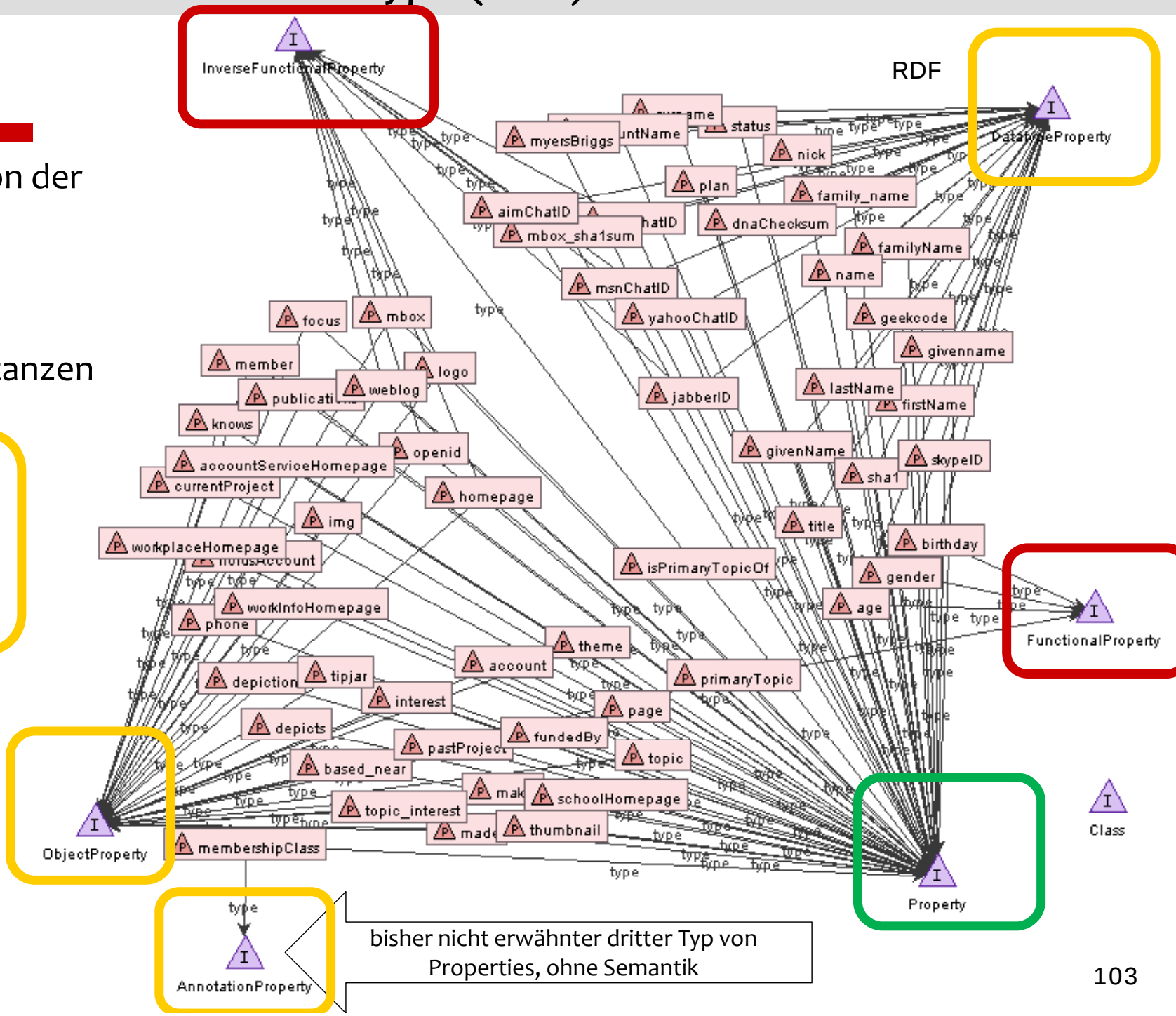
Aus OWL kommen die beschreibenden Instanzen für die Relationen:

Art der Rolle:

- ObjectProperty
- DataTypeProperty
- AnnotationProperty

Rollen-Eigenschaften:

- FunctionalProperty
- …

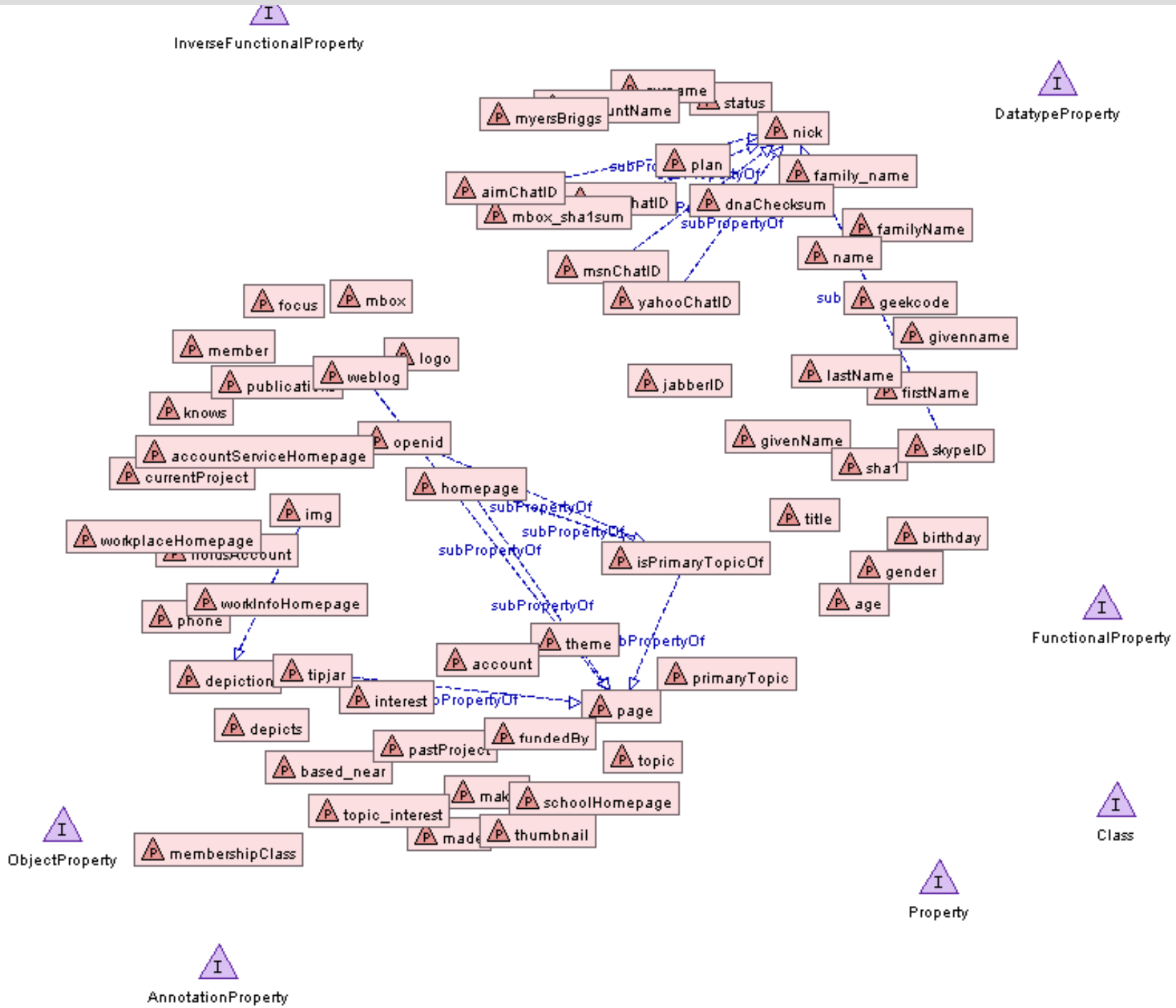


bisher nicht erwähnter dritter Typ von Properties, ohne Semantik

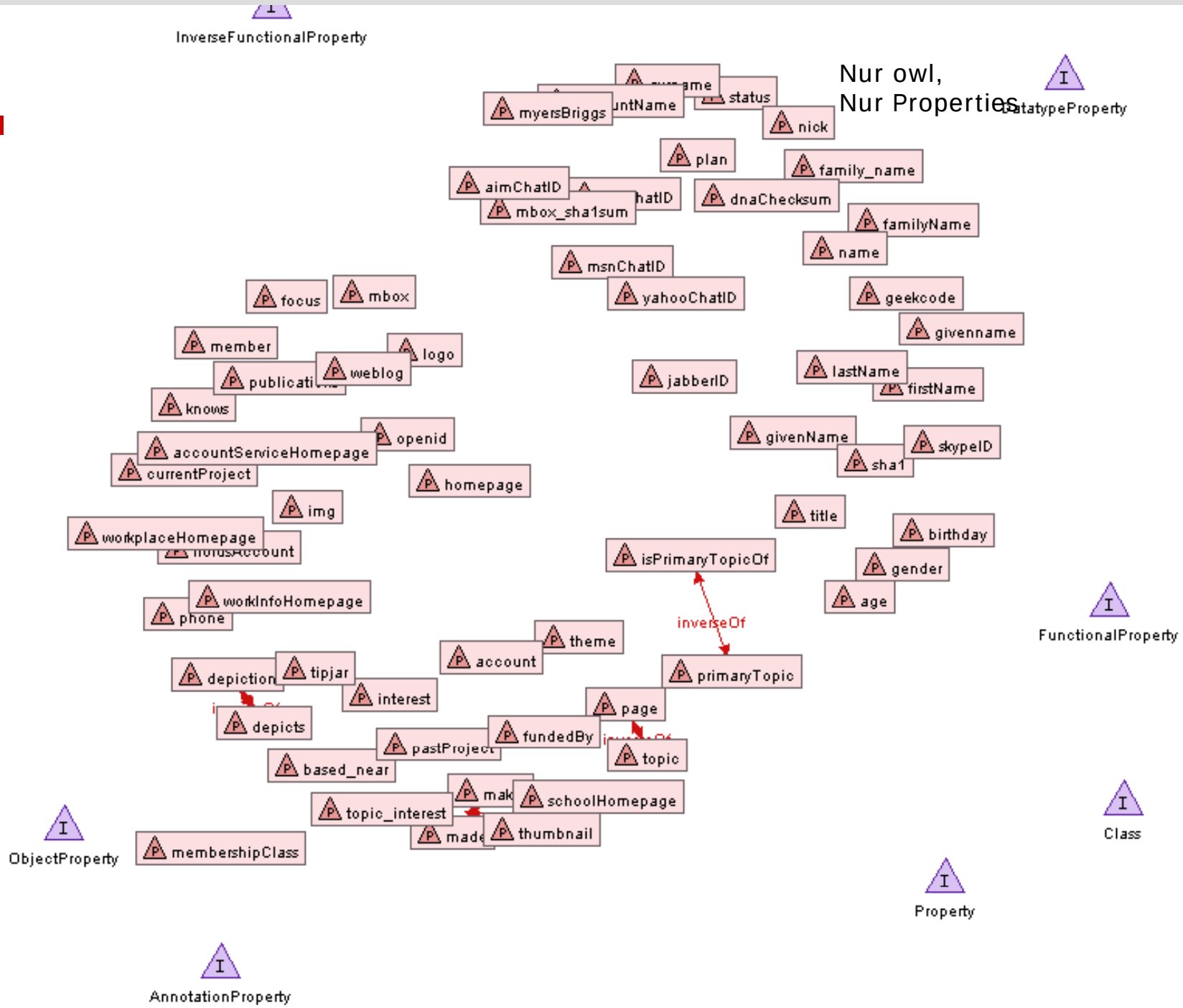
FOAF: nur Properties und Instanzen: subPropertyOf (RDFS)

rdfs:subPropertyOf =

- Spezifizierung von Properties ergibt die **Rollenhierarchie**



FOAF: nur Properties und Instanzen: inverseOf (OWL)



- Sind Sie bereit für einen RDF-Auszug aus FOAF?

Repräsentation in RDF: Auszug aus FOAF

```
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Organization" rdfs:label="Organization"
      rdfs:comment="An organization." vs:term_status="stable">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"></rdf:type>
      <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"></rdfs:subClassOf>
      <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"></rdfs:isDefinedBy>
      <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Person">  </owl:disjointWith>
      <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Document"> </owl:disjointWith>
</rdfs:Class>
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Group" vs:term_status="stable"
      rdfs:label="Group" rdfs:comment="A class of Agents.">
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"></rdf:type>
     <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"></rdfs:subClassOf>
</rdfs:Class>
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Agent" vs:term_status="stable"
      rdfs:label="Agent" rdfs:comment="An agent (eg. person, group, software or physical
      artifact).">
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"></rdf:type>
     <owl:equivalentClass rdf:resource="http://purl.org/dc/terms/Agent"></owl:equivalentClass>
</rdfs:Class>
```

Vernetzung von Ontologien

- Zusammenfassung der OWL-Sprachelemente

OWL Sprachelemente I

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:strong und abgeleitete
- xsd:decimal und abgeleitete, wie xsd:integer
- xsd:float, xsd:double
- xsd:boolean, xsd:dateTime, xsd:time,

OWL Sprachelemente II

Klassenkonstruktoren

- owl:Thing
- owl:Nothing
- owl:oneOf
- owl:intersectionOf
- owl:unionOf
- owl:complementOf
- **Rollenrestriktionen!**
  - owl:allValuesFrom
  - owl:someValuesFrom
  - owl:hasValue
  - owl:cardinality
  - owl:minCardinality
  - owl:maxCardinality

Axiome über Klassen

- **owl:equivalentClass**
- **rdfs:subClassOf**
- owl:disjointWith
- owl:Class

OWL Sprachelemente III

Rollenkonstruktoren, -beziehungen und –eigenschaften

- owl:ObjectProperty
- owl:DatatypeProperty
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
- owl:equivalentProperty
- owl:inverseOf
- owl:TransitiveProperty
- owl:SymmetricProperty
- owl:FunctionalProperty
- owl:InverseFunctionalProperty

Next

- Ontologie und die beiden Visionen des ‚Semantic Web'
- RDF, RDFS
- Web Ontology Language OWL
  - OWL lesen – Sprachkonstrukte
  - Beispiel einer Ontologie: FOAF
- Von OWL zu OWL2
- Anfragen an Ontologien
  - OWL-Reasoner sind zurückhaltend (Open World Assumption)

OWL 2

- W3C-Standard seit Oktober 2009
- **OWL 2 ist entscheidbar**

PR

Von SHOIN(D) zu SROIQ(D)

1. H -> R Rollenhierarchie auch komplexer Rollen $hatFreund \circ hatFeind \sqsubseteq hatFeind$
   - Bsp: Die Feinde meiner Freunde sind auch meine Feinde
2. N -> Q qualifizierte Zahlenrestriktion $Person \sqcap \geq 3\, hatKind.Professor$

3. Negierte Rollen in der Abox: ‚paul ist nicht Vater von anna‘ $\neg R(a, b)$
4. Self-Konzept ‚Ich mag mich‘
5. neue Rollenaxiome: reflexiv, irreflexiv, antisymmetrisch
6. Disjunkte Rollen disjunkt(hatVater, hatSohn)
7. Universelle Rolle (topObjectProperty)

Überblick $\mathcal{SROIQ}$

Neu in OWL 2

Klassenausdrücke

Klassennamen	A, B
Konjunktion	$C \sqcap D$
Disjunktion	$C \sqcup D$
Negation	$\neg C$
Exist. Rollenrestr.	$\exists R.C$
Univ. Rollenrestr.	$\forall R.C$
4. Self	$\exists S.\mathrm{Self}$
2. Größer-als	$\geq n\ S.C$
Kleiner-als	$\leq n\ S.C$
Nominale	$\{a\}$

Rollen

Rollennamen	R, S, T
einfache Rollen	S, T
Inverse Rollen	R^-
7. Universelle Rolle	U

TBox (Klassenaxiome)

Inklusion	$C \sqsubseteq D$
Äquivalenz	$C \equiv D$

RBox (Rollenaxiome)

Inklusion	$R_1 \sqsubseteq R_2$
Allgemeine Inkl.	$R_1^{(-)} \circ \ldots \circ R_n^{(-)} \sqsubseteq R$ 1.
Transitivität	$\mathrm{Tra}(R)$
Symmetrie	$\mathrm{Sym}(R)$
Reflexivität	$\mathrm{Ref}(R)$ 5.
Irreflexivität	$\mathrm{Irr}(S)$
Disjunktheit	$\mathrm{Dis}(S, T)$ 6.

ABox (Fakten)

Klassenzugehörigkeit	$C(a)$
Rollenbeziehung	$R(a, b)$
Neg. Rollenbeziehung	$\neg S(a, b)$ 3.
Gleichheit	$a \approx b$
Ungleichheit	$a \not\approx b$

Wichtige Fragmente von OWL 2

- **OWL 2 EL**
  - beruht auf EL++
    - Nur Existenzquantor, Konjunktion, Nominale, konkrete Datentypen, Bottom, Top, Konzept- und Rollensubsumption
  - Vorteil: polynomielle Komplexität
  - Anwendung: SNOMED-CT, NCI Thesaurus, Gene Ontology, OpenGalen ...
  - Freier Reasoner: CEL (A polynomial-time Classifier for the description logic EL+), TU Dresden

- OWL 2 QL, beruht auf $\mathrm{DL}\text{-}lite_{\mathcal{R}}$
- OWL 2 RL
- OWL 2 FULL

Next

- Ontologie und die beiden Visionen des ‚Semantic Web'
- RDF, RDFS
- Web Ontology Language OWL
  - OWL lesen – Sprachkonstrukte
  - Beispiel einer Ontologie: FOAF
- Von OWL zu OWL2
- Anfragen an Ontologien
  - OWL-Reasoner sind zurückhaltend (Open World Assumption)
- Ontologien in den Life Sciences
  - GALEN, SNOMED

Anfragen an Ontologien

- Die TBox und die ABox beinhalten implizites Wissen – dieses soll explizit und damit für einen Benutzer sichtbar gemacht werden

Anfragen an Ontologien

- Wissensbasis einer DL besteht aus TBox und ABox

- OWL-Full ist eine Teilmenge der Prädikatenlogik 1. Stufe (PL1, FOL)
- Diese ist unentscheidbar, d.h. die Inferenzalgorithmen terminieren nicht immer

- Wie löst OWL-DL dieses Problem?

Einschänkung der Ausdruckskraft führt zur Entscheidbarkeit:

- OWL Lite ist entscheidbar, *SHIF*(D)
- OWL-DL ist entscheidbar, *SHOIN(D)*
- OWL 2 ist entscheidbar, *SROIQ(D)*

Anfragen an Ontologien

- **Beispiel**
  - Anfrage an OWL-DL-Wissensbasis: **Ist C eine Unterklasse von D?**
  - Sie bekommen auf jeden Fall eine Antwort in endlicher Zeit. (Entscheidbarkeit)
  - Was bedeutet: Antwort = JA
  - Was bedeutet: Antwort = NEIN

Open World Assumption

- **Bei Anfragen an DL gilt die Open World Assumption**
  - open world: Das Wissen der Wissensbasis über die Welt ist unvollständig.
  - Was nicht beweisbar ist, ist nicht automatisch ?????
  - Auch wenn es keine Erklärung gibt, kann eine Aussage ?????
  - Anfrage (bspw. nach Instanzen) liefert **nur beweisbare Antworten**

- **Closed World Assumption (z.B. in Prolog)**
  - Annahme: Alles Wissen ist in der Wissensbasis enthalten.
  - Also: Was nicht aus der Wissensbasis beweisbar ist, ist falsch.

Anfragen an Ontologien

PR

- **Beispiel**
  - Anfrage an OWL-DL-Wissensbasis: **Ist C eine Unterklasse von D?**
  - Sie bekommen auf jeden Fall eine Antwort in endlicher Zeit. (Entscheidbarkeit)
  - Was bedeutet: Antwort = JA
  - Was bedeutet: Antwort = NEIN

- Was bedeutet: Antwort = JA
  - Es lässt sich beweisen, dass C eine Unterklasse von D ist.
- Was bedeutet: Antwort = NEIN
  - Es lässt sich nicht beweisen, dass C eine Unterklasse von D ist
  - Die Annahme C sei keine Unterklasse von D führt nicht zu einem Widerspruch.
  - Es lässt sich nicht beweisen, dass C eine Unterklasse von D ist – aber auch nicht ausschließen.
  - Also versteht der Reasoner die Frage so: ***„Kannst Du beweisen, dass C eine Unterklasse von C ist?“***

PR

TBox

Beispiel1 Open World Assumption – die TBox

c) Formulieren Sie die beschreibungslogischen Ausdrücke zu den Axiomen über die Konzepte A und B in folgendem OWL-Dokument:

```
<owl:Class rdf:about="A">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="hatKind"/>
            <owl:allValuesFrom rdf:resource="Weiblich"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="B">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="hatKind"/>
            <owl:someValuesFrom rdf:resource="Weiblich"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

$$A \equiv \forall hatKind.Weiblich$$

$$B \sqsubseteq \exists hatKind.Weiblich$$

Beispiel1 Open World Assumption – die ABox

d) Über die Instanzen ist Folgendes bekannt:

```
<owl:Thing rdf:about="andrea"></owl:Thing>
<owl:Thing rdf:about="fred"></owl:Thing>
<owl:Thing rdf:about="lisa"><rdf:type rdf:resource="Weiblich"/></owl:Thing>
<owl:Thing rdf:about="paul">
    <hatKind rdf:resource="andrea"/>
    <hatKind rdf:resource="lisa"/>
</owl:Thing>
```

$Thing(andrea)$

$Thing(fred)$

$Thing(lisa)$

$Weiblich(lisa)$

$Thing(paul)$

$hatKind(paul, andrea)$

$hatKind(paul, lisa)$

PR

Beispiel1 Open World Assumption – die Anfragen

Begründen Sie, ob sich folgende Beziehungen mit der TBox aus c) ableiten lassen:

`A(paul)`	Leicht
`A(fred)`	Schwierig
`B(paul)`	Schwierig
`B(fred)`	Leicht

TBox

$$A \equiv \forall hatKind.Weiblich$$

$$B \sqsubseteq \exists hatKind.Weiblich$$

ABox

$Thing(andrea)$

$Thing(fred)$

$Thing(lisa)$

$Weiblich(lisa)$

$Thing(paul)$

$hatKind(paul, andrea)$

$hatKind(paul, lisa)$

PR

Beispiel1 Open World Assumption – die Antworten

A(paul): **Nein**. Wäre Weiblich(andrea) noch bekannt, wäre die Antwort ebenfalls ‚nein'.

A(fred): **Nein**. Zwar gilt für alle bekannten Kinder (das sind null) von fred, dass sie weiblich sind, aber es können unbekannte Söhne existieren.

B(paul): **Nein**. Paul gehört zwar zum Konzept derer, die mindestens ein weibliches Kind haben (rechte Seite der Inklusion), von dem das Konzept B eine Unterklasse ist. Ob er aber zum Konzept B gehört, ist unbekannt. B könnte bspw. das Konzept der Personen mit zwei klugen Töchtern sein.

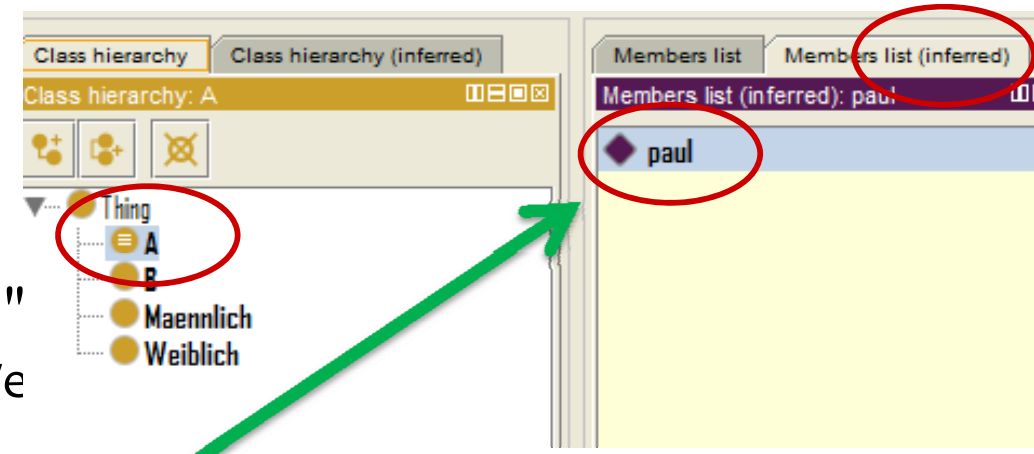
B(fred): **Nein**: Ob fred überhaupt eine Tochter hat, ist nicht bekannt, und über die Zugehörigkeit zu B ebenfalls nichts. Er könnte aber durchaus zu B gehören.

Das Antworten bei OWA sind erkennbar zurückhaltend, es wird nichts Falsches behauptet. Das hat den großen Vorteil, dass das System bei Erweiterung der Wissensbasis (neues Wissen) seine alten Antworten niemals widerrufen muss.

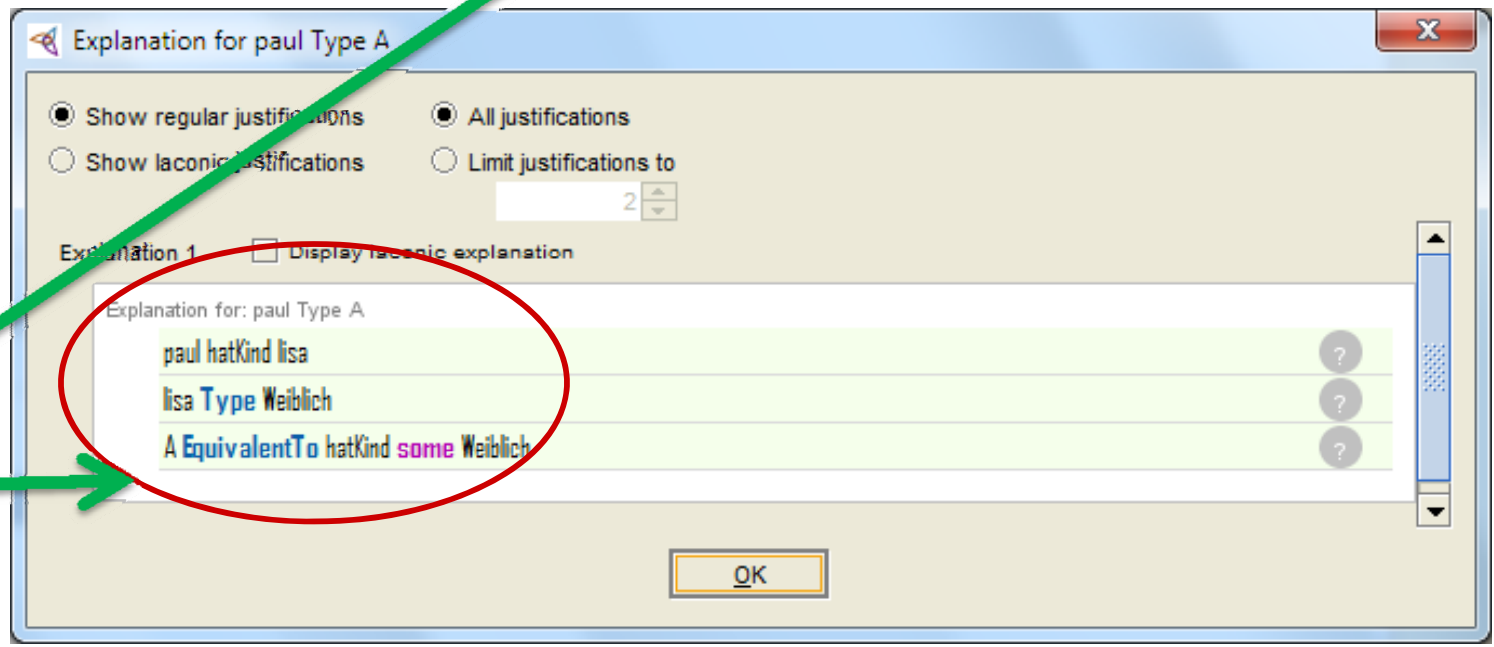
Für eine positive Antwort ändern wir Konzept A

```
<owl:Class rdf:about="A">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource=„hatKind"
      <owl:someValuesFrom rdf:resource="We
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

PR



Wenn es für A genügt, nur ein weibliches Kind zu haben, dann **gehört paul zu A** – und das System kann es **erklären:**



Beispiel2 Open World Assumption – TBox und Frage

$VegetarianPizza \equiv Pizza \sqcap \forall hasTopping.(CheeseTopping \sqcup VegetableTopping)$

$MargheritaPizza \sqsubseteq NamedPizza \sqcap \exists hasTopping.MozzarellaTopping \sqcap \exists hasTopping.TomatoTopping$

$TomatoTopping \sqsubseteq VegetableTopping$

$MozzarellaTopping \sqsubseteq CheeseTopping$

Frage: $MargheritaPizza \sqsubseteq VegetarianPizza?$?

Beispiel2 Open World Assumption – Antwort

$VegetarianPizza \equiv Pizza \sqcap \forall hasTopping.(CheeseTopping \sqcup VegetableTopping)$

$MargheritaPizza \sqsubseteq NamedPizza \sqcap \exists hasTopping.MozzarellaTopping \sqcap \exists hasTopping.TomatoTopping$

$TomatoTopping \sqsubseteq VegetableTopping$

$MozzarellaTopping \sqsubseteq CheeseTopping$

Frage: $MargheritaPizza \sqsubseteq VegetarianPizza?$?

- **Nein.** Solange nicht gesagt wird, dass eine MargheritaPizza **nur** die beiden Toppings Tomate und Mozzarella haben kann, könnte sie auch andere, fleischhaltige Toppings haben und ist somit nicht automatisch eine vegetarische Pizza
- Heilung durch **Closure** (Angabe aller Möglichkeiten) von hasTopping:

$$MargheritaPizza \sqsubseteq NamedPizza... \sqcap \forall hasTopping.(MozzarellaTopping \sqcup TomatoTopping)$$

- **Was wollen Sie eine Ontologie fragen?**
  - 6 Anfragen an eine TBox
  - 5 Anfragen an eine TBox mit ABox

6 Terminologische Anfragen an OWL – nur TBox

1. TBox-Konsistenz (Erfüllbarkeit):
   - Enthält die Begriffswelt Widersprüche?

2. Klassenkonsistenz (concept satisfiability)
   - Kann es Individuen in diesem Konzept geben?
   - Oder ist C äquivalent zu owl:Nothing

3. Klassenäquivalenz (concept equivalence)
   - Unerwünschte Synonyme und Redundanz

- Todo Zuordnungsübung Formel zu Anfrage

6 Terminologische Anfragen an OWL – nur TBox

4. Klassendisjunktheit

5. Subklassenbeziehung (concept subsumption)
   - Ist C aus D folgerbar?

6. Klassifikation (classifier)
   - Erzeugt alle Subklassenbeziehungen, und damit die Hierarchierelation (*inferred* ontology class hierarchy)

- Todo Zuordnungsübung Formel zu Anfrage

5 Assertionale Anfragen an OWL –TBox und ABox

1. ABox-Konsistenz
   - Haben ABox und TBox ein gemeinsames Modell?

2. Instanzüberprüfung (instance checking)
   - Gehört eine Instanz a zur Klasse C?
3. Realisierung (realizer)
   - Zu welchen Konzepten gehört eine Instanz?

4. Werden zwei gegebene Individuen durch Rolle verknüpft?
   - Ist anton der reiche Großonkel von frieda?

5 Assertionale Anfragen an OWL –TBox und ABox

5. Instanzgenerierung, Retrieval
   - Suche nach allen Individuen, die in einer Klasse enthalten sind
   - Finde alle Filler für eine Rolle zu einer Instanz
   - Finde alle Rollen zwischen zwei Instanzen
   - Finde alle Instanzpaare zu einer Rolle.

OWL-Werkzeuge

Editoren

- **Protegé**, freier Ontologieeditor, Plugins/Erweiterungen, http://protege.stanford.edu
- SWOOP, open source, http://code.google.com/p/swoop/
- OWL Tools, http://owltools.ontoware.org/
- Ontostudio, kommerzieller Editor
- TopBraid Composer, kommerzieller Editor
- NeOn-Toolkit, freier Editor mit kommerziellen Erweiterungen

Inferenzmaschinen, Reasoner

- **HermiT**, http://hermit-reasoner.com/
- **Pellet,** http://clarkparsia.com/pellet/
- KAON2, http://kaon2.semanticweb.org
- **FACT++,** http://owl.cs.manchester.ac.uk/fact++/
- Racer, http://www.racer-systems.com/
- Cerebra, http://www.cerebra.com/

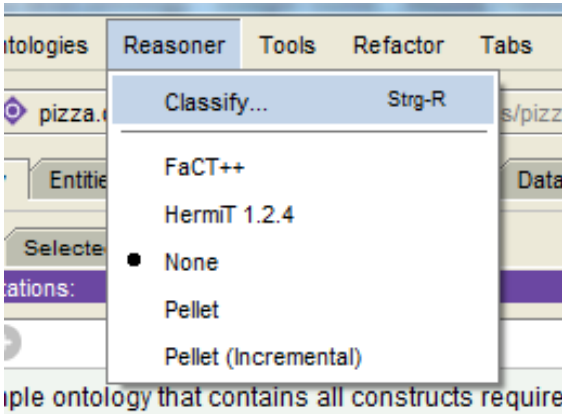
RDF-Datenbanken (RDF-Stores)

- Jena, Oracle 10g, RAP, Redland, Sesame, Virtuoso, . . .

- Aktuelle DL-Reasoner unter: http://www.cs.man.ac.uk/~sattler/reasoners.html

Reasoner in Protege

- Consistency checking
  - check whether or not it is possible for the class to have any instancesPrimitive
- Subsumption testing
  - test whether or not one class is a subclass of another class



- Automated Classification ‚Classify…‘
  - compute the **inferred** ontology class **hierarchy**
  - the reasoner can **only** automatically classify classes under **defined classes**
  - Inferred role hierarchy
  - Instance classification (->Members der Klassen)

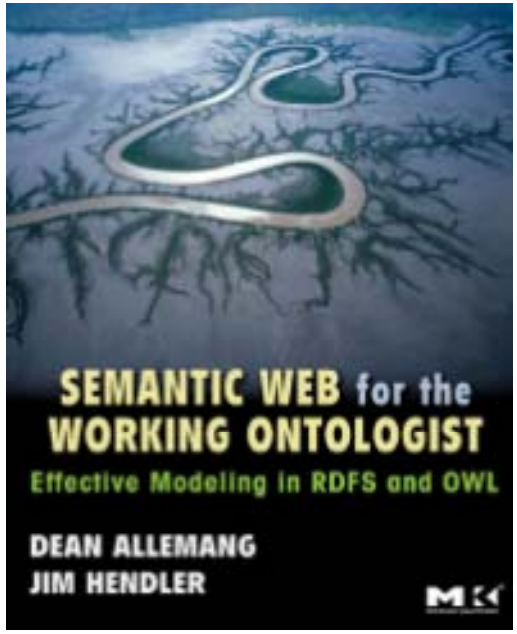
-> Übung

Vorgehen bei der Erstellung von Ontologien

1. Bestimmen von Fachgebiet, Anwendungsziel und Umfang der Ontologie.
2. **Recherche und Bewertung zur eventuellen Wiederverwendung bereits bestehender Ontologien.**
3. Sammeln der für das Gebiet wichtigen Begriffe.
4. Definieren von Klassen und ihren Hierarchien.
5. Definieren von Rollen und Merkmalen
6. Definieren von Rollen- und Merkmalsfüllern
7. Erzeugen von Instanzen.

[nach SS08, Kap11]

- Weiter in: Semantic Web for the Working Ontologist
- OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns



Next

- Ontologie und die beiden Visionen des ‚Semantic Web'
- RDF, RDFS
- Web Ontology Language OWL
  - OWL lesen – Sprachkonstrukte
  - Beispiel einer Ontologie: FOAF
- Von OWL zu OWL2
- Anfragen an Ontologien
  - OWL-Reasoner sind zurückhaltend (Open World Assumption)
- Ontologien in den Life Sciences
  - GALEN, SNOMED
- Wertung DL

Domänenontologie

- **top-level ontology:**
  - allgemeine, bereichsübergreifende Ontologien zur Beschreibung von bspw. Raum, Zeit, fundamentaler Rollen
- **domain ontology:**
  - grundlegende Begriffswelt eines Diskursbereiches / Domäne, bspw. Medizin
- **task ontology:**
  - grundlegende Begriffswelt einer Aufgabe, eines Problembereiches, bspw. Fotografieren

-> Domänenontologien in der Medizin

Examples of **OWL** ontology applications in Life Science

- Open Biomedical Ontologies Consortium, see http://obo.sourceforge.net
  - **Gene Ontology** (GO)
  - **Microarray Gene Expression Data** (MGED) ontology.
- US National Cancer Institute (NCI), see http://ncit.nci.nih.gov/
  - **NCI Thesaurus** (NCIt) - Terminologie der Krebsbehandlung, 50.000Klassen, EL++
- **EHROntology**, see http://trajano.us.es/~isabel/EHR/
  - An electronic health record ontology based on openEHR work
- **Neurolex**, Neuroscience Lexicon mit Semantic MediaWiki, http://neurolex.org

- **OpenGALEN**
  - European Galen project, part of which has involved the construction of a large DL Knowledge Base describing medical terminology

Anwendungen komplexer Ontologien in Medizin/Biologie

- **Gene Ontology**
  - Domäne: Gene in allen Organismen
  - Ontologiesprache: „OBO“, Abbildung auf OWL teilweise möglich, **EL++**
  - Anwendung: Kombination von Gen-Daten
  - 25.000 Klassen

- **SNOMED-CT – Systematized Nomenclature of Medicine (Clinical Terms)**
  - Domäne: Krankheiten, Diagnosen, Medikamente, . . .
  - Ontologiesprache: Beschreibungslogik **EL++** (Fragment von OWL 2)
  - Anwendung: Informationsaustausch in Medizinanwendungen

- **OpenGALEN**
  - Domäne: Krankheiten (medizinische Klassifikation)
  - Ontologiesprache OWL 2.0 , (überwiegend **EL++**)
  - Anwendung?: „basis for teaching, training and services“ (Zitat)
  - Formale Semantik hilft bei Ontologieerstellung

- Galen

DL in der Medizin: Galen

- **G**eneral **A**rchitecture for **L**anguages, **E**nclopedias and **N**omenclatures in Medicine
- Entwicklungsbeginn Anfang der 90er Jahre, seit 2000 openSource (OpenGALEN)
- http://www.opengalen.org
- Europäisches Forschungsprojekt, Ontologie als OpenGALEN vergügbar
- Menschenlesbare Repräsentation: Sprache GRAIL, abbildbar in OWL

- Testfeld für OWL-Tools wie Reasoner oder Editoren
- 83 OWL ontology components, Wurzelfile OpenGALEN8_FULL.owl
- 4 Hauptkomponenten
  - *Open*GALEN Common Reference Model (CRM) version 8
  - The Diseases Extension
  - The Surgical Procedures Extension
  - The Drug Ontology Extension

Galen - Metrik

- DL expressivity ALERIF(D)

ALERIF(D)

- AL = Attributive Language (atomare Negation, Schnitt, AllR.c, exR.Top)
- E = exR.C (volle existentielle Restriktion)
- R = Rolleninklusion komplexer Rollen
- I = inverse Rollen
- F = Funktionale Properties
- (D) konkrete Domain

Metrics

- **Class: 125391 !**
- **Object property: 990**
- Data property: 63
- Individual: 29

Class Axioms

- **SubClassOf axioms: 53449**
- **EquivalentClasses axioms: 113622 !**
- DisjointClasses axioms: 0
- Hidden GCI: 7892

Object Property Axioms

- **SubObjectPropertyOf axioms count 998**
- EquivalentObjectProperties axioms: 14
- InverseObjectProperties axioms: 488
- FunctionalObjectProperty axioms: 332
- SubPropertyChainOf axioms: 385

- SNOMED CT

DL in der Medizin: SNOMED CT

PR

- SNOMED CT basiert auf der Beschreibungslogik EL++

  - Natürliche Sprache

    Jede Hepatitis ist eine Entzündung, die in einer Leber lokalisiert ist.
    Jede Entzündung in einer Leber ist eine Hepatitis.

  - Prädikatenlogik

    $\forall x$ $instanceOf(x, Hepatitis) \rightarrow instanceOf(x, Inflammation) \wedge$
    $\exists y: instanceOf(y, Liver) \wedge hasLocation(x,y)$

  - Beschreibungslogik

    $Hepatitis \equiv Inflammation \sqcap \exists hasLocation.Liver$

[Ing2010]

SNOMED CT: Description Logic (Baader, Dresden)

DL-Dialekt EL:

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

DL-Dialekt EL+:

weitere Rollen-Axiome:

general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Tabelle 4: $\mathcal{EL}^{++}$

Konstruktor	Syntax	Semantik
Bottom	$\bot$	$\emptyset$
concrete domain	$p(f_1 \ldots f_n)$ for $p \in P^{D_j}$	$\{d \mid$es existieren $e_1 \ldots e_k \in \Delta^{D_j}$: $f_i^{\mathcal{I}}(d) = e_i$, for $1 \leqslant i \leqslant k$ und $(e_1 \ldots e_k) \in p^{D_j}\}$

übeck / 24

[Ing2010]

DL-Dialekt „EL+“

Beispiel:

$$\text{Pericardium} \sqsubseteq \text{Tissue} \sqcap \exists\text{contained-in}.\text{Heart}$$

$$\text{Pericarditis} \sqsubseteq \text{Inflammation} \sqcap \exists\text{has-location}.\text{Pericardium}$$

$$\text{Inflammation} \sqsubseteq \text{Disease} \sqcap \exists\text{acts-on}.\text{Tissue}$$

$$\text{Heartdisease} \doteq \text{Disease} \sqcap \exists\text{has-location}.\text{Heart}$$

$$\text{Heartdisease} \sqsubseteq \exists\text{has-state}.\text{NeedsTreatment}$$ GCI

$$\text{has-location} \circ \text{contained-in} \sqsubseteq \text{has-location}$$ Role Inclusion

d.h. $\text{Pericarditis} \sqsubseteq \text{Heartdisease}$

$\sqsubseteq \exists\text{has-state}.\text{NeedsTreatment}$

Ingenerf, Lübeck / 25

[Ing2010]

SNOMED CT: Beispiel

CONCEPT: 397825006 Gastric ulcer

DESCRIPTORS (TERMS):
Gastric ulcer (1777426014, preferred), Stomach ulcer (1785985013),
GU - Gastric ulcer (1785986014), Gastric ulceration (1785987017)

Defining concepts instead of creating hierarchies manually:

DEFINITION:
Fully defined by ...
- *Is a* 64572001 Disease (disorder)
- Group
  - *Associated morphology* 56208002 Ulcer
  - *Finding site* 69695003 Stomach structure

using description logic (DL) formalism.

Creating subsumption-hierarchy automatically by a classifier:

INFERRED SUPERORDINATE CONCEPTS (by a classifier)
- 29384001 Disorder of stomach
- 40845000 Gastrointestinal ulcer

Ingenerf, Lübeck (19)

SNOMED CT: Beispiel (Definieren statt hierarchisieren)

Descriptions
- F 2394452012 Helicobacter-assoziiertes pylorisches Ulkus (Störung)
- P 2394453019 Helicobacter-assoziiertes pylorisches Ulkus

Fully defined by ...
- 116680003 ist ein/e
  - D 128070006 infektiöse Krankheiten des Abdomens
  - D 312121001 bakterielle Infektionskrankheit des Magen-Darm-Trakts
  - D 39204006 pylorusnahes Magengeschwür
  - D 6185008 in Verbindung mit Helicobacter stehende Krankheit
- 246075003 verursachendes Agens
  - D 80774000 helicobacter pylori
- Group
  - 116676008 assoziierte Morphologie
    - D 56208002 Geschwür
  - 363698007 Befundlokalisation
    - D 280119005 Struktur des Magenpförtners

Beschreibungslogik
Können Sie es lesen?

- Die Begriffshierarchie wird rechnergestützt deduziert.

Fazit

- Was leisten formale Ontologien?
- Was leisten formale Ontologien nicht?

Was leisten formale Ontologien?

Weltsicht durch die mengentheoretische Brille

Repräsentation

- Exakte, logikbasierte Beschreibungen von Konzepten und Relationen, die durch konkrete Objekte der Welt instanziiert werden (Konzeptualisierung)

Vernetzung

- Einigung über kontrollierte (mächtige und vernetzte) Vokabularien
  - Vernetzung verteilter Wissensbasen
  - Konsistenz von Begriffen

Inferenz

- Verwendung von maschinellem Schließen, z.B. basierend auf Beschreibungslogiken (OWL-DL, OWL 2)
- Automatisches Einordnen neuer Konzepte, Automatisches Erstellen von Hierarchien, Konsistenzprüfung, Erfüllbarkeit

Was leisten formale Ontologien nicht?

- Keine Repräsentation
  - Unsicheren Wissens (Wahrscheinlichkeiten, Sicherheiten)
  - Unscharfen Wissens (Fuzzy)
  - Prozeduralen Wissen (Regeln, RIF, SWRL)

- Vollständigkeit und Konsistenz nur in begrenzten Wissensgebieten

Semantische Webtechnologien sind der Anfang, das menschliche Wissen in großem Umfang für Maschinen zu erschließen.

Zusammenfassung

- *Eine **Ontologie** ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung.*
  - *in OWL: Menge von Konzeptdefinitionen (…) einer Deskriptionslogik*
  - *Vision formaler Ontologien: Vernetzung und Inferenz*
  - *Wissensbasis: TBox und ABox*
- *Semantische Ausdruckskraft von RDF: R(b,c) und C(a), ABox*
- *RDFS: Klassen, Rollen, Klassen- und Rollenhierarchie, Range und Domain*

- ***OWL**: OWL Full, OWL DL, OWL Lite, Entscheidbarkeit*
  - *OWL DL: SHOIN(D)*
  - *OWL-Dokument in RDF/XML-Syntax*
  - *Dl in OWL: Konzept- und Rollenkonstruktoren, Axiome über Rollen, Konzepte, Instanzen*

- *OWL 2: SROIQ(D)*
- *Anfragen an OWL-Ontologien, OWA*
- *Beispiele: FOAF, GALEN, SNOMED CT*

Quellen

[Gaus2005] Gaus, W. Dokumentations- und Ordnungslehre: Theorie und Praxis des Information Retrieval Springer, Berlin, 2005

[Ing2010] Josef Ingenerf SNOMED CT - Terminologische Fundierung für Medizinische Wissensbasen, Workshop „Wissensbasierte Systemkomponenten in Krankenversorgung und Lehre, conhIT „Satellitenveranstaltung“, Berlin, 19. April 2010

[SS08] C. Spreckelsen and K. Spitzer, Wissensbasen und Expertensysteme in der Medizin: KI-Ansätze zwischen klinischer Entscheidungsunterstützung und medizinischem Wissensmanagement, Vieweg+Teubner, 2008.

[Win88] Chaffin, Roger; Herrmann, Douglas J.; Winston, Morton. "An empirical taxonomy of part-whole relations: Effects of part-whole relation type on relation identification" Language and Cognitive Processes 3.1, 1988

[Zai02] Albrecht Zaiß, Bernd Graubner, Josef Ingenerf, Florian Leiner, Ulrich Lochmann, Michael Schopen, Ulrich Schrader und Stefan Schulz: Medizinische Dokumentation, Terminologie und Linguistik, in Handbuch der Medizinischen Informatik, Hrsg. von Thomas Lehmann und Erdmuthe Meyer zu Bexten, ISBN 3-446-21589-1, 2002

Und andere im Text genannte