Wirtschaftsinformatik II – Meilicke/Stuckenschmidt

Modellierungsmuster, Umsetzung in Protege

ONTOLOGIEN
MODELLIERUNGSMUSTER

Übersicht

Wie erstellt man eine Ontologie, die sich aus den folgenden Arten von Axiomen zusammensetzt (richtige Anwendung, Motivation, in Protege, …)

- Konzepthierarchie
  - $Company \sqsubseteq Organisation$
- Disjunktheit
  - $Person \sqsubseteq \neg Organisation$
- Einschränkung Domain/Range
  - $\exists worksFor.\top \sqsubseteq Person$
  - $\exists\, worksFor^{-1}.\top \sqsubseteq Company$
- Einschränkung Konzepte über Rollen
  - $Employee \sqsubseteq \exists worksFor.\top$
  - $Company \sqsubseteq \exists worksFor^{-1}.\top$
- Rollenhierarchie
  - $writes \sqsubseteq creates$
- Inverse Rollen
  - $createdBy^{-1} \equiv creates$
- Eigenschaften von Rollen (oder Attributen)
  - $func(marriedTo)$

Konzepthierarchie

- Die Konzepthierarchie wird durch eine Menge von Axiomen der Form $A \sqsubseteq B$ definiert

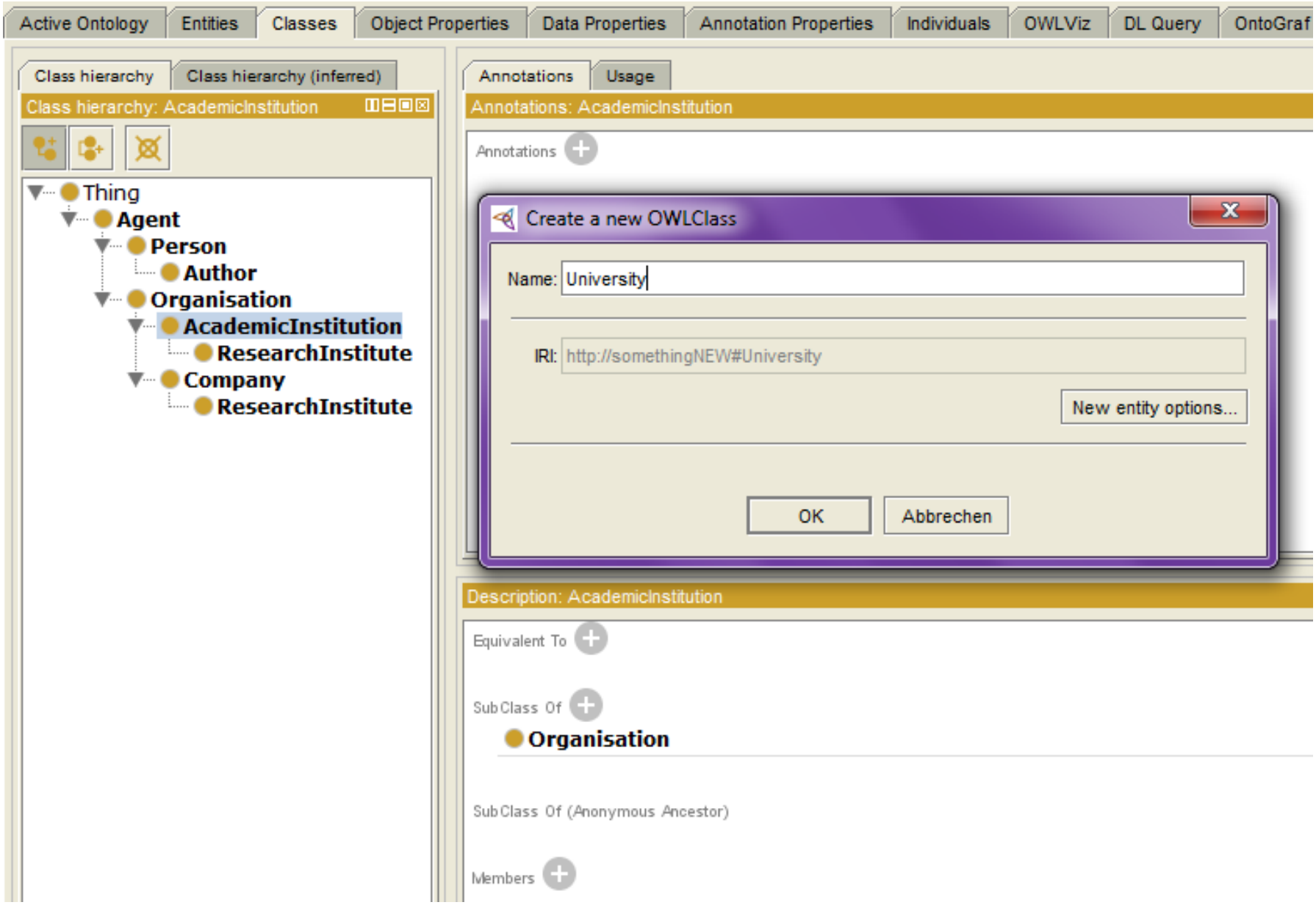
- In vielen Fällen wird dadurch ein Baum erzeugt, so dass jedes Konzept nur genau ein direktes Superkonzept hat

- **Agent**
  - **Organisation**
    - **Company**
    - **AcademicInstitution**
      - **University**
      - **ResearchInstitute**
  - **Person**
    - **Author**
    - **Painter**

Konzepthierarchie

- Aber: es kann auch Sinn machen, dass ein Konzept mehrere direkte Superkonzepte hat
  - Mehrfachvererbung macht auch beim Programmieren Sinn
  - Eine solche Modellierung ist nicht falsch

- **Agent**
  - **Organisation**
    - **Company**
      - **ResearchInstitute**
    - **AcademicInstitution**
      - **University**
      - **ResearchInstitute**
  - **Person**
    - **Author**
    - **Painter**


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: AcademicInstitution
Thing
Agent
Person
Author
Organisation
AcademicInstitution
ResearchInstitute
Company
ResearchInstitute
Annotations
Usage
Annotations: AcademicInstitution
Annotations
Create a new OWLClass
Name: University
IRI: http://somethingNEW#University
New entity options...
OK
Abbrechen
Description: AcademicInstitution
Equivalent To
SubClass Of
Organisation
SubClass Of (Anonymous Ancestor)
Members

Konzepthierarchie

- “Beliebte” Fehler beim Modellieren der Konzepthierarchie sehen so aus

- **Europe**
  - **Germany**
    - **Berlin**
    - **Mannheim**
  - **France**
    - **Paris**
- **Asia**
  - **China**
    - **Peking**
  - **Japan**
    - **Tokio**

- **EmployeeType**
  - **Mangager**
    - **TopManager**
  - **Developer**
  - **Administrator**
- **CompanyType**
  - **...**

- **ProgrammCommittee**
  - **ExternalReviewer**
  - **JuniorMember**
  - **SeniorMember**
  - **PCChair**
- **LocalCommittee**
  - **...**

Konzepthierarchie

- Wenn zwei Konzepte nebeneinander stehen, d.h. Subkonzepte desselben Superkonzepts sind, dann sind sie nicht zwangsläufig disjunkt (“disjoint”)

- **Agent**
  - **Organisation**
    - **Company**
      - **ResearchInstitute**
    - **AcademicInstitution**
      - **University**
      - **ResearchInstitute**
  - **Person**
    - **Author**
    - **Painter**

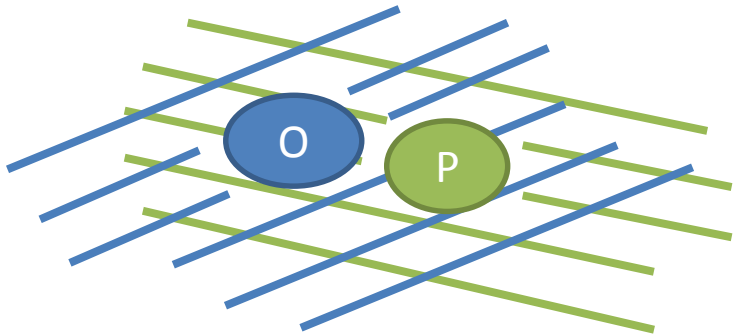
- In diesem Fall kann es durchaus Instanzen geben, die sowohl *Company* als auch *AcademicInstitution* sind

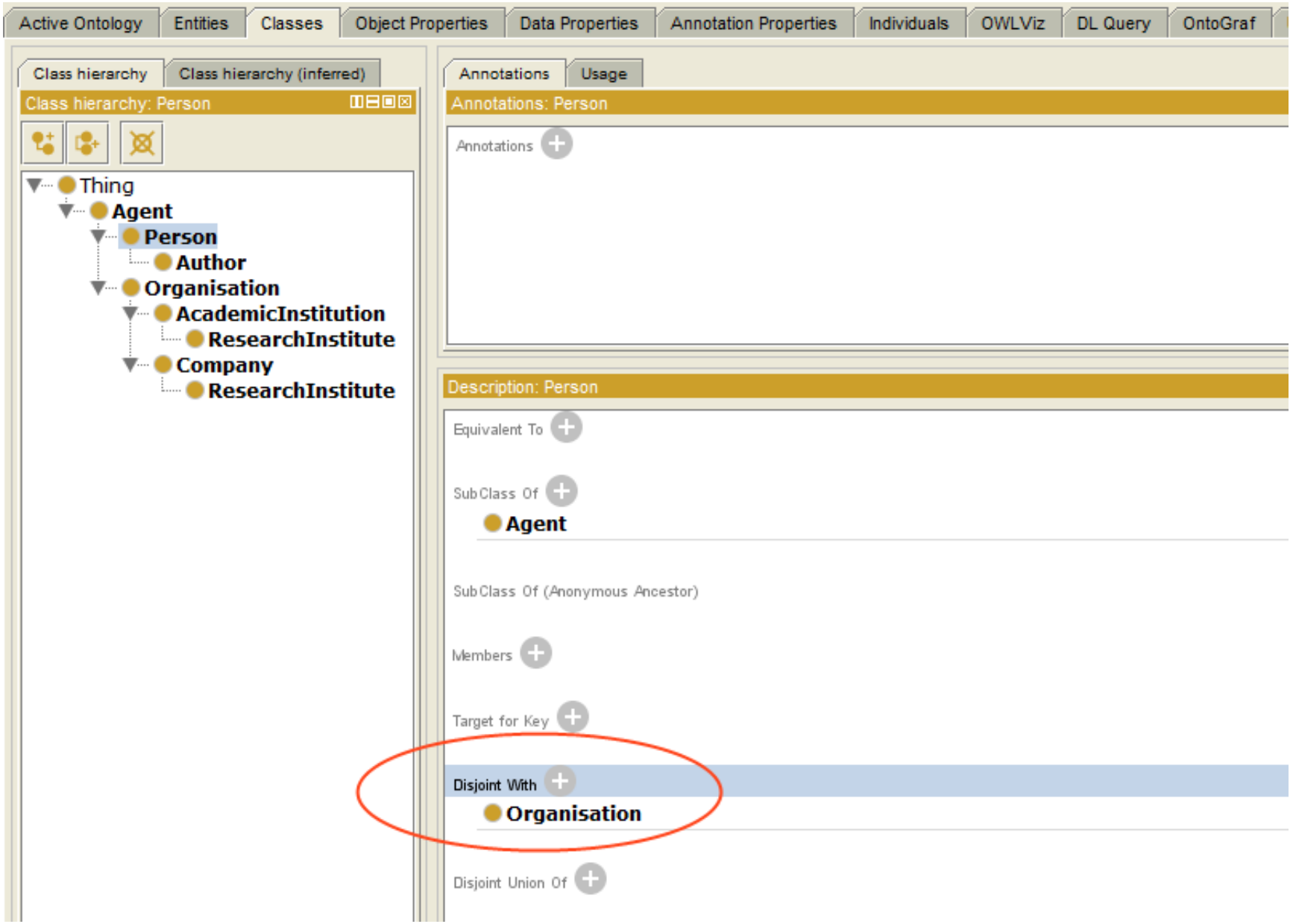
Konzepthierarchie

- Disjunktheit von Konzepten muss man explizit mittels Axiomen spezifizieren
  - $Organisation \sqsubseteq \neg Person,\ University \sqsubseteq \neg ResearchInstitute$

- Achtung:
  - $Organisation \sqsubseteq \neg Person$ ist äquivalent zu $Person \sqsubseteq \neg Organisation$
  - $\neg Organisation \sqsubseteq Person$ hat eine andere Behauptung

- **Agent**
  - **Organisation**
    - **Company**
      - **ResearchInstitute**
    - **AcademicInstitution**
      - **University**
      - **ResearchInstitute**
  - **Person**
    - **Author**




Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Person
Thing
Agent
Person
Author
Organisation
AcademicInstitution
ResearchInstitute
Company
ResearchInstitute
Annotations
Usage
Annotations: Person
Annotations
Description: Person
Equivalent To
SubClass Of
Agent
SubClass Of (Anonymous Ancestor)
Members
Target for Key
Disjoint With
Organisation
Disjoint Union Of

Rollen und Attribute

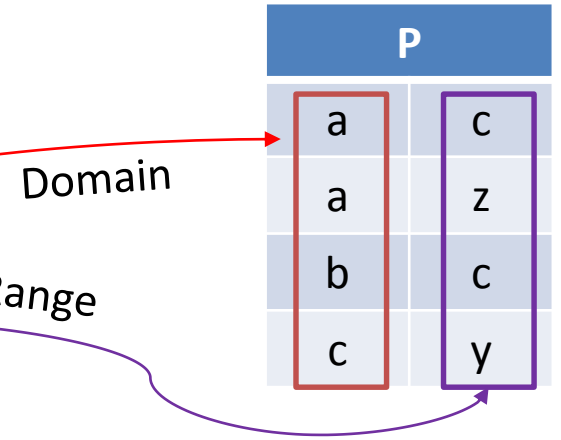
- Identifiziere Beziehungen zwischen Instanzen der bisher eingeführten Konzepte
  - Eine Person arbeitet für eine Company => Person worksFor Company
- Identifiziere, was man zusätzlich aussagen will, was aber keine Beziehung zwischen Instanzen sein kann
  - Eine Person hat einen Namen => Persson -> hasName -> String
- Attribute vs. Erweiterung der Konzepthierarchie
  - Eine Person hat eine Webseite
    - Person -> hasWebpage -> String (Attribut)
    - Person -> hasWebpage -> Webpage
  - Überlege, ob man auch Aussagen über die Webseite treffen möchte!

Rollen

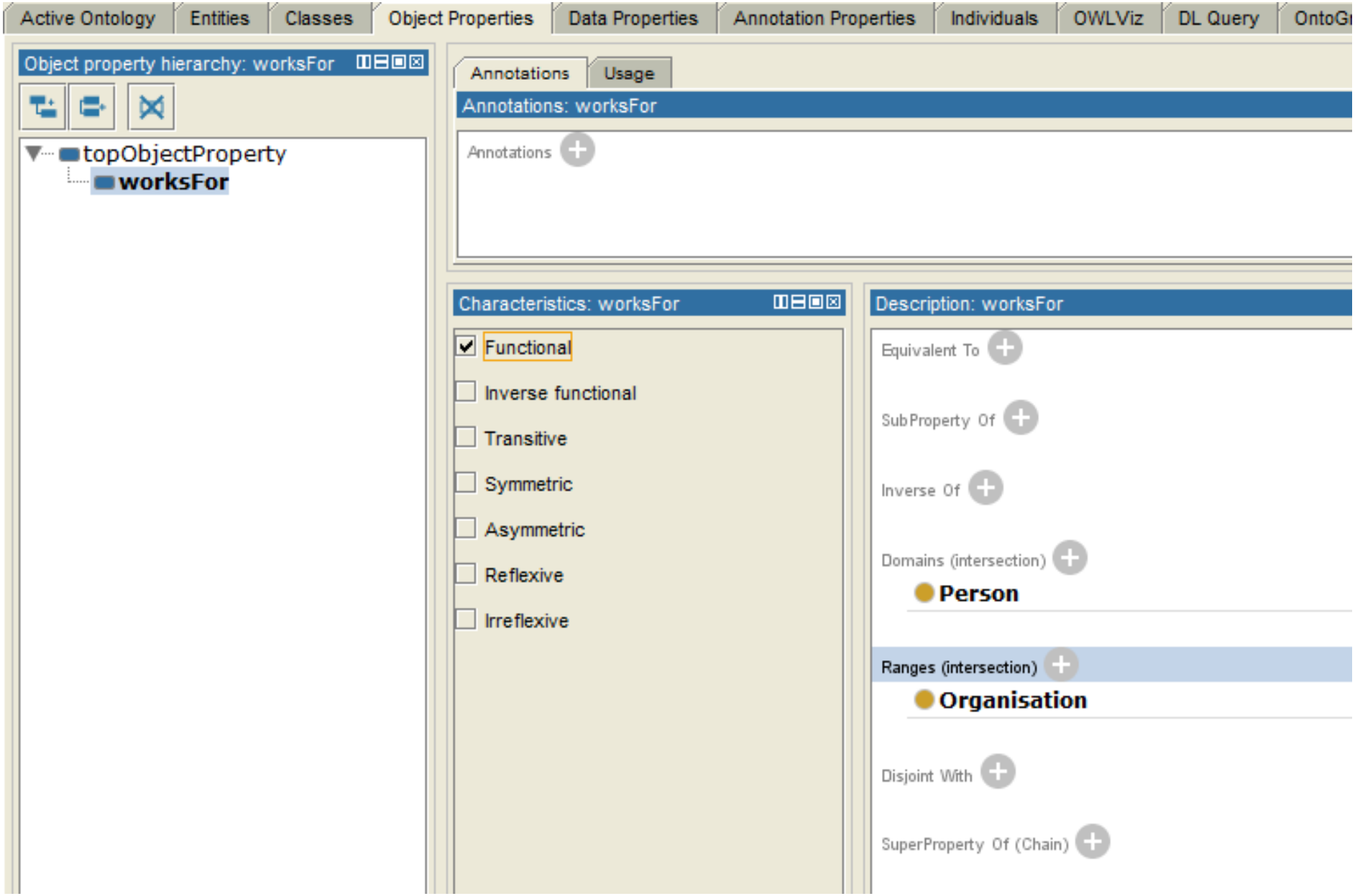
- Was kann überhaupt nur in der Beziehung stehen, die durch die Rolle P(x,y) beschrieben wird

- Man nennt die Menge der x-Entitäten, die Domain der Rolle, und die Menge der y-Entitäten die Range der Rolle

- Domain und Range kann man mittels der folgenden Axiome einschränken
  - $\exists P.\top \sqsubseteq X$
  - $\exists\, P^{-1}.\top \sqsubseteq Y$ (oder: $\top \sqsubseteq \forall P.Y$)



  - $\exists worksFor.\top \sqsubseteq Person$
  - $\exists\, worksFor^{-1}.\top \sqsubseteq Company$ (oder: $\top \sqsubseteq \forall worksFor.Company$)


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
Object property hierarchy: worksFor
topObjectProperty
worksFor
Annotations
Usage
Annotations: worksFor
Annotations
Characteristics: worksFor
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: worksFor
Equivalent To
SubProperty Of
Inverse Of
Domains (intersection)
Person
Ranges (intersection)
Organisation
Disjoint With
SuperProperty Of (Chain)

Rollen

- Muss eine Instanz eines Konzepts in einer Rollen-Beziehung zu etwas stehen, wenn es eine Instanz diese Konzepts ist?
  - Ein Mitarbeiter arbeitet mindestens für eine Firma
  - Eine Firma hat mindestens einen Mitarbeiter

- Wenn ja, kann man dies durch die folgenden Axiome ausdrücken
  - $Employee \sqsubseteq \exists worksFor.\top$
    - Wenn man noch spezifischer sein will: $EmployeeIT \sqsubseteq \exists \mathrm{worksFor}.ITCompany$
  - $Company \sqsubseteq \exists worksFor^{-1}.\top$
    - Analog: $ITCompany \sqsubseteq \exists \mathrm{worksFor}^{-1}.EmployeeIT$

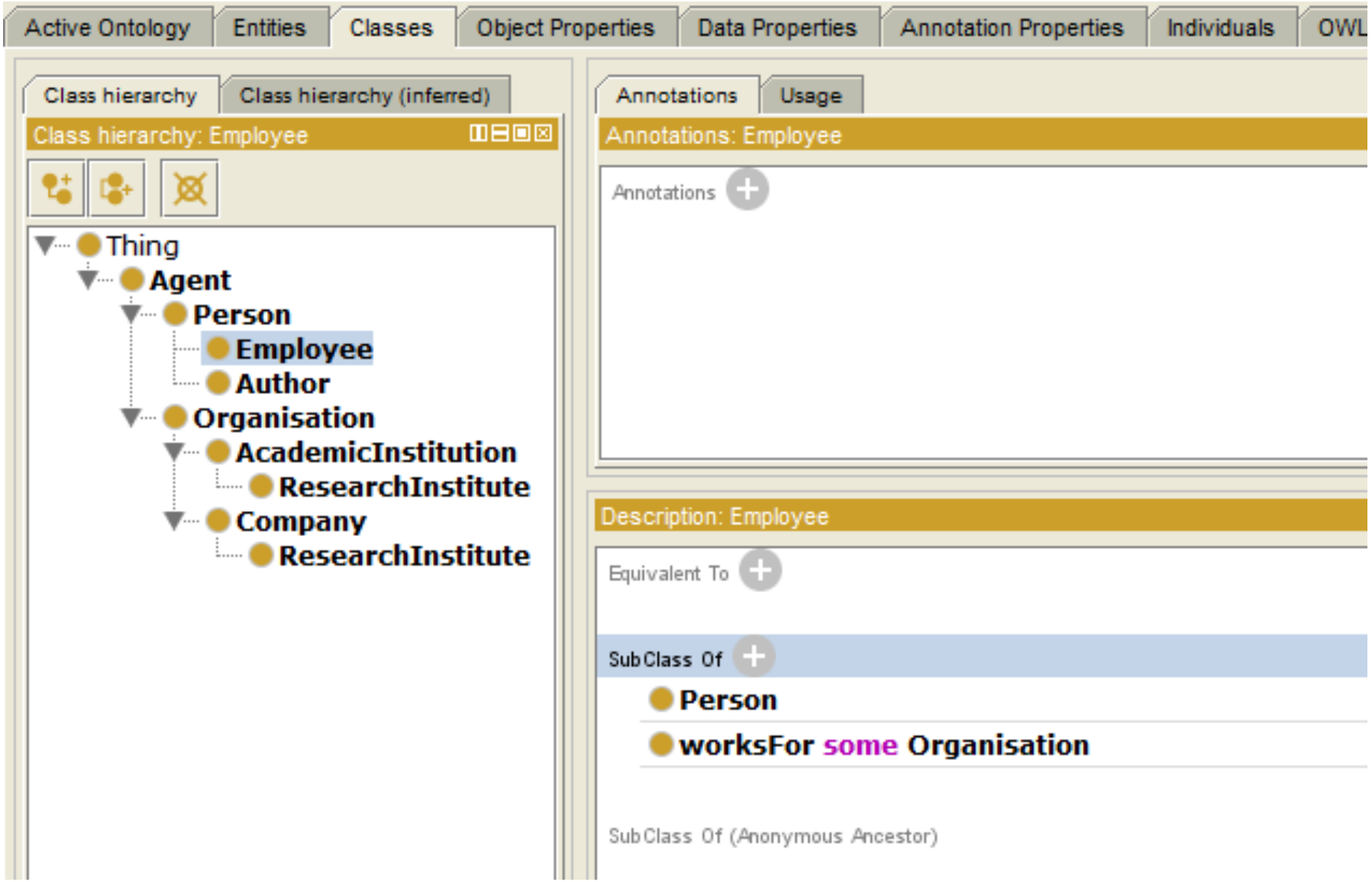
- Achtung: Richtung der Subsumption ist genau umgekehrt im Vergleich zur Domain/Range Einschränkung

Manchester Syntax

- Einige Formeln kann man in Protege nicht „zusammenklicken“, da zu komplex
  - Formeln müssen getippt werden, aber Quantoren und Junktoren gibt es nicht auf der Tastatur

- Einfache Repräsentation mittels Manchester Syntax

Formal	Manchester
$University \sqcap Company$	$University\ and\ Company$
$University \sqcup Company$	$University\ or\ Company$
$\exists worksFor.Company$	$worksFor\ some\ Company$
$\forall worksFor.Company$	$worksFor\ only\ Company$


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWL
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Employee
Thing
Agent
Person
Employee
Author
Organisation
AcademicInstitution
ResearchInstitute
Company
ResearchInstitute
Annotations
Usage
Annotations: Employee
Annotations
Description: Employee
Equivalent To
SubClass Of
Person
worksFor some Organisation
SubClass Of (Anonymous Ancestor)

Rollen

- Rollen können auch in einer Hierarchie angeordnet werden
- Eine Rolle S ist eine Superrolle einer Rolle R genau dann, wenn jedes Paar das in der Beziehung R zueinander steht auch in der Beziehung S zueinander steht
- Besipiele
  - $worksAsCEOFor \sqsubseteq worksFor$
  - $writes \sqsubseteq creates$
  - $paints \sqsubseteq creates$

Rollen

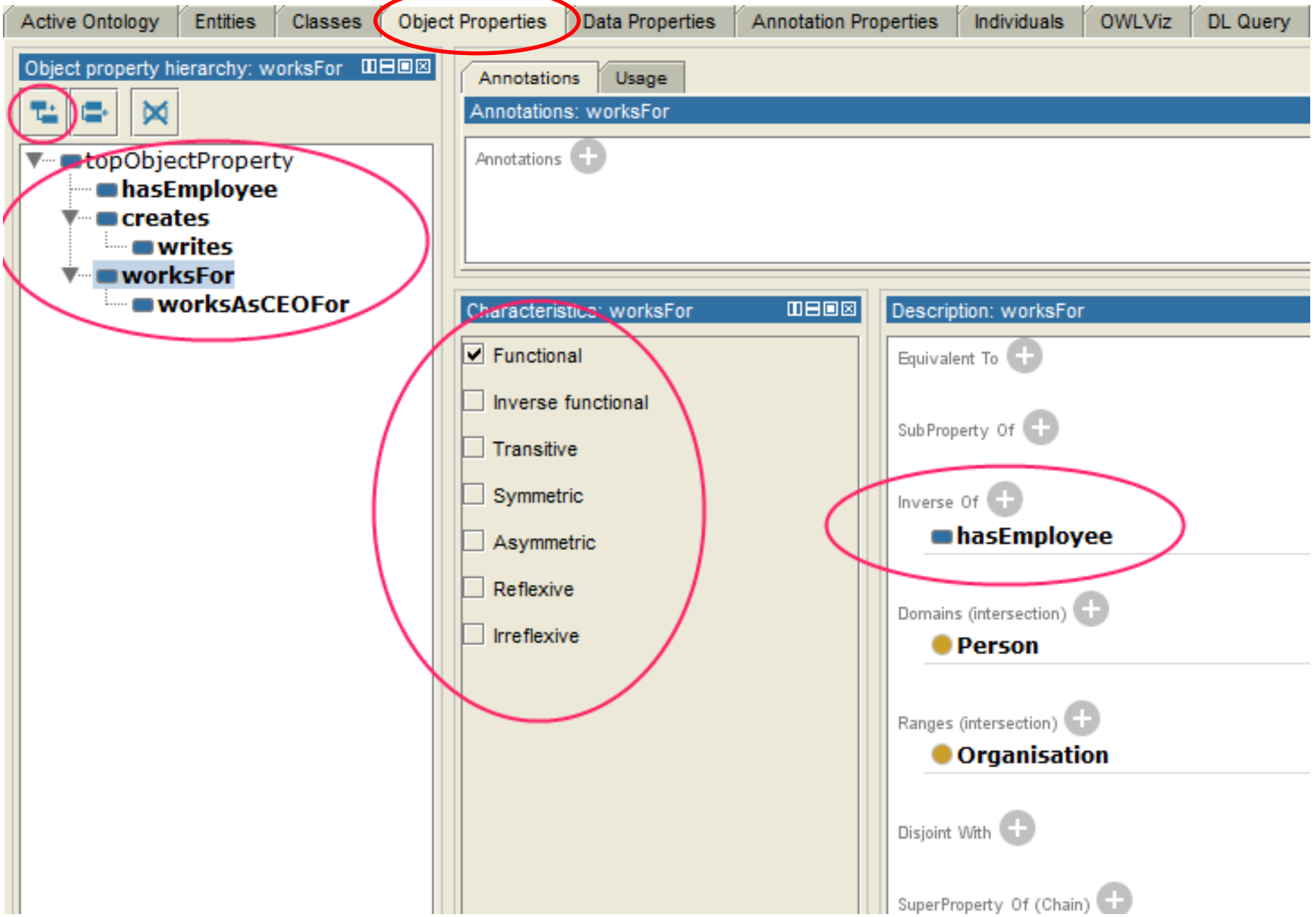
- Man kann inverse Rollen definieren
- Eine Rolle R’ ist eine inverse Rolle zu R genau dann, wenn für jedes <x,y> das in der Beziehung R zueinander steht <y,x> in der Beziehung R’ zueinander steht
- Beispiele
  - $hasEmployee \equiv worksFor^{-1}$ (äquiv. $hasEmployee^{-1} \equiv worksFor$)
  - $createdBy \equiv creates^{-1}$
- Wird vor allem eingeführt um Lesbarkeit zu erhöhen und andere Definitionen einfacher zu machen

Rollen

- Formale Eigenschaften von Rollen sollten angegeben werden, um
  - bestimmte Sachverhalte zu verbieten
  - neues ableitbar zu machen

- Beispiele
  - *func(marriedTo)*
  - *func(firstAuthor)*
  - *sym(marriedTo)*
  - *trans(partOf)*

- Funktionalität kann auch für Attribute spezifiziert werden
  - func(hasFirstName)
  - func(birthDate)


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Individuals
OWLViz
DL Query
Object property hierarchy: worksFor
topObjectProperty
hasEmployee
creates
writes
worksFor
worksAsCEOFor
Annotations
Usage
Annotations: worksFor
Annotations
Characteristics: worksFor
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: worksFor
Equivalent To
SubProperty Of
Inverse Of
hasEmployee
Domains (intersection)
Person
Ranges (intersection)
Organisation
Disjoint With
SuperProperty Of (Chain)

Attribute

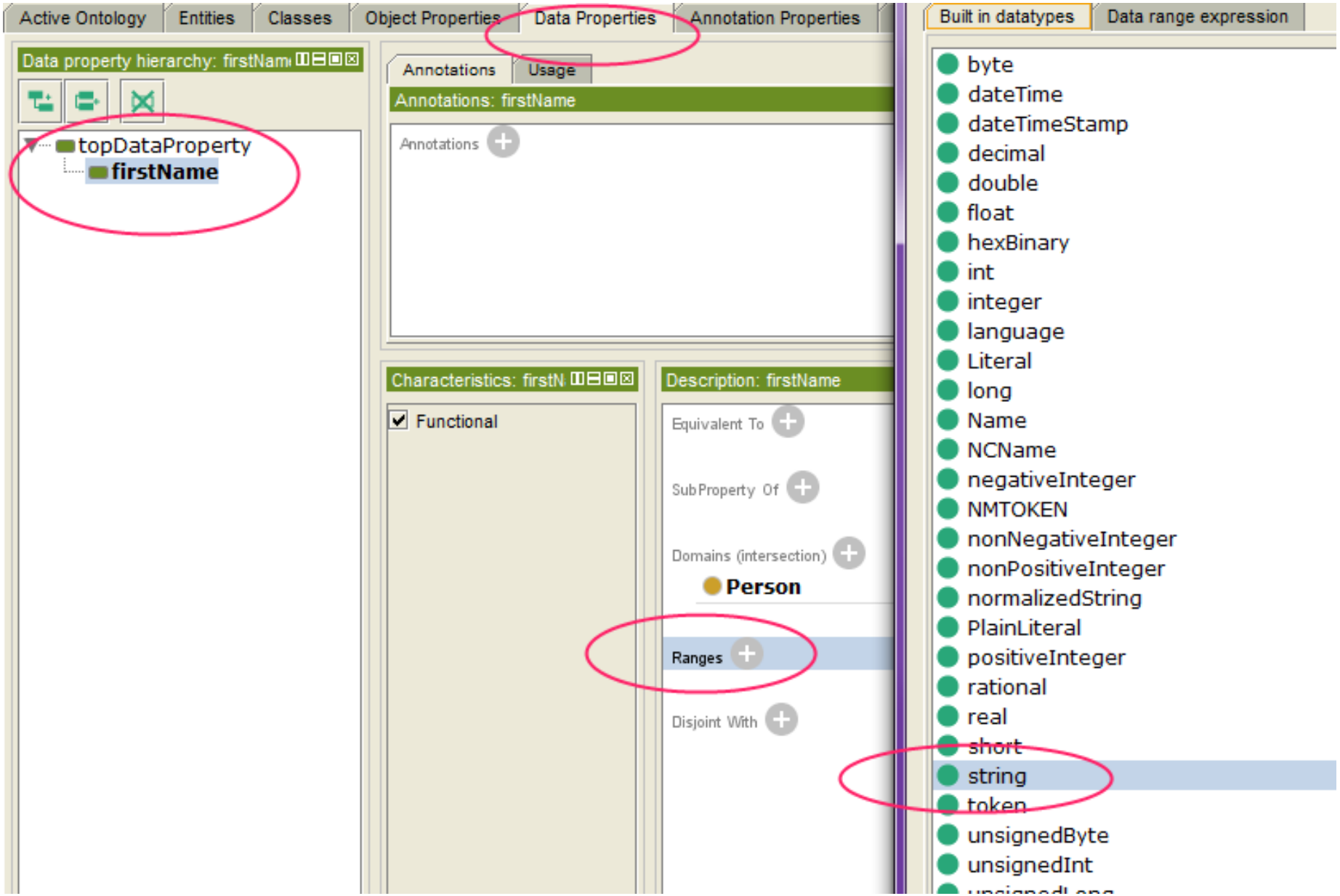
- Attribute benötigen einen Datentyp, der die Range des Attributs einschränkt

- Als Datentypen stehen die im XML Standard definierten Datentypen zur Verfügung
  - String, DateTime, Integer, ...

P	
a	0.2
a	4
b	4.1
c	-2

- Attribute können funktional sein, aber z.B. nicht symmetrisch oder transitiv (macht auch keinen Sinn)

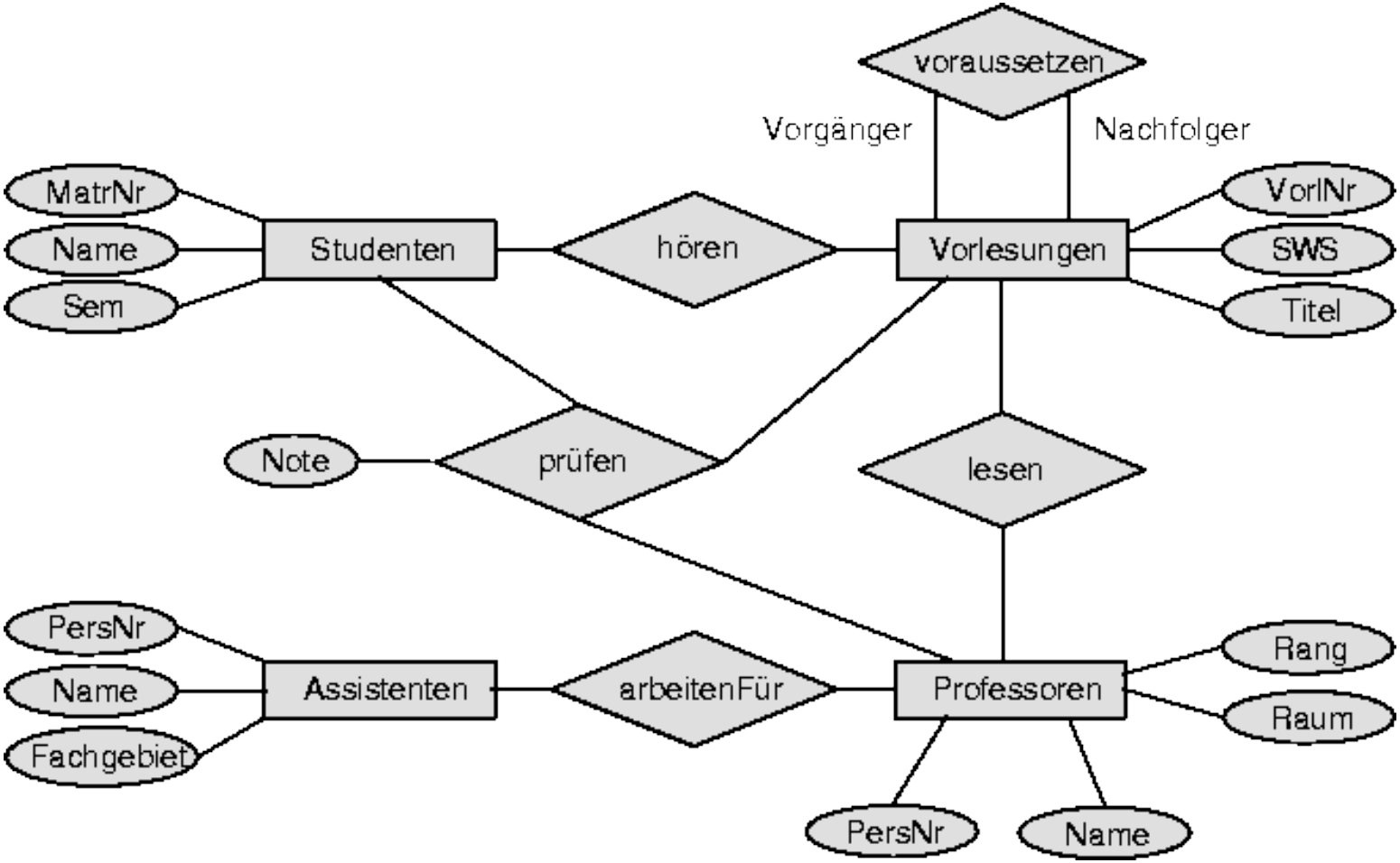
- Eine Ontologie ohne Attribute macht nur wenig Sinn, wenn man Aussagen über einen Teilbereich der Welt treffen möchte


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Data property hierarchy: firstNam
topDataProperty
firstName
Annotations
Usage
Annotations: firstName
Annotations
Characteristics: firstN
Functional
Description: firstName
Equivalent To
SubProperty Of
Domains (intersection)
Person
Ranges
Disjoint With
Built in datatypes
Data range expression
byte
dateTime
dateTimeStamp
decimal
double
float
hexBinary
int
integer
language
Literal
long
Name
NCName
negativeInteger
NMTOKEN
nonNegativeInteger
nonPositiveInteger
normalizedString
PlainLiteral
positiveInteger
rational
real
short
string
token
unsignedByte
unsignedInt

Übersicht

- Konzepthierarchie
  - $Company \sqsubseteq Organisation$
- Disjunktheit
  - $Person \sqsubseteq \neg Organisation$
- Einschränkung Domain/Range
  - $\exists worksFor.\top \sqsubseteq Person$
  - $\exists\, worksFor^{-1}.\top \sqsubseteq Company$
- Einschränkung Konzepte über Rollen
  - $Employee \sqsubseteq \exists worksFor.\top$
  - $Company \sqsubseteq \exists worksFor^{-1}.\top$
- Rollenhierarchie
  - $writes \sqsubseteq creates$
- Inverse Rollen
  - $createdBy^{-1} \equiv creates$
- Eigenschaften von Rollen (oder Attributen)
  - $func(marriedTo)$

Zusammenhang zu ER-Diagramm



Beschreibungslogik vs. OWL

- Beschreibungslogik ist formale Grundlage von OWL

- OWL ist ein syntaktischer Standard mit dem Beschreibungslogik ausdrückbar ist
  - Kann in verschiedenen Formaten gespeichert werden (z.B. XML basiert)
  - Web Ontology Language
  - Protege ist ein Editor für OWL Ontologien

Beschreibungslogik	OWL
Konzept	Class
Rolle	Objectproperty
Attribut	Data(type)property
Instanz, Individuum	Individual

Zusammenfassung & Vorausschau

- Typische Modellierungsmuster
  - Wie sich diese als Axiome umsetzen lassen
  - Wie man diese Axiome in Protege erstellt

- Letzter Foliensatz, in dem es um den Gesamtprozess der Entwicklung einer Ontologie geht