

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

Aufgabe 1

Entwickeln Sie ein RDF-Schema für ein Bibliotheksinformationssystem.

Eine Bibliothek besitzt Bücher. Bibliotheken haben einen Namen, eine Adresse und eine Telefonnummer. Bücher haben einen Titel, einen oder mehrere Autoren, und eine ISBN-Nummer. Personen haben einen Namen, eine Adresse, eine Telefonnummer und eine E-Mailadresse. Bücher können von einer Person entliehen sein.

Verwenden Sie für Ihre Lösung möglichst bestehende Schemata wieder, z.B. FOAF und Dublin Core:

- <http://xmlns.com/foaf/spec/20100809.rdf>
- <http://dublincore.org/2010/10/11/dcterms.rdf>

Aufgabe 1

- Sammeln wir erst einmal die Klassen

*Eine **Bibliothek** besitzt **Bücher**. Bibliotheken haben einen Namen, eine Adresse und eine Telefonnummer. Bücher haben einen Titel, einen oder mehrere **Autoren**, und eine ISBN-Nummer. **Personen** haben einen Namen, eine Adresse, eine Telefonnummer und eine E-Mailadresse. Bücher können von einer Person entliehen sein.*

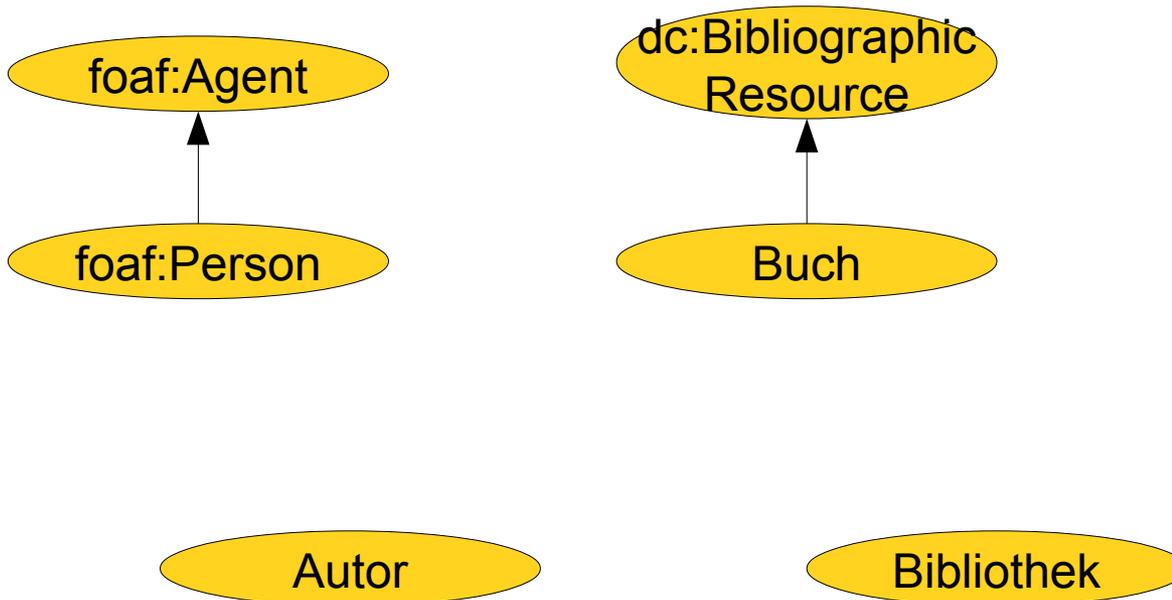
Aufgabe 1

- Nachschauen, was in den wiederverwendbaren Schemata schon gibt
 - Bibliothek: –
 - Buch: `dc:BibliographicResource`

```
<rdfs:comment xml:lang="en-US">  
A book, article, or other documentary resource.  
</rdfs:comment>
```
 - Autor: –
 - Person: `foaf:Person`

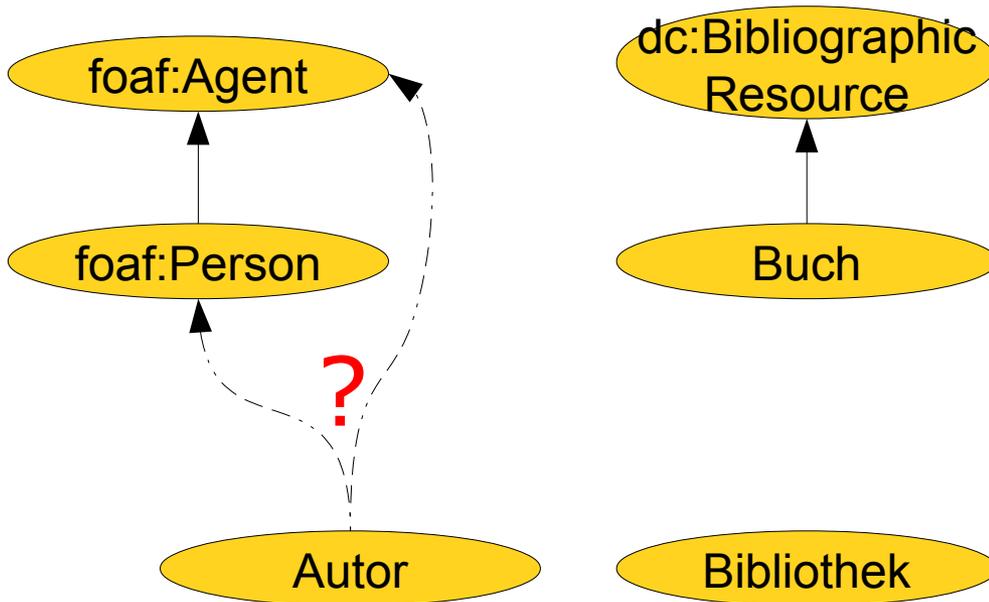
Aufgabe 1

- Klassen anordnen



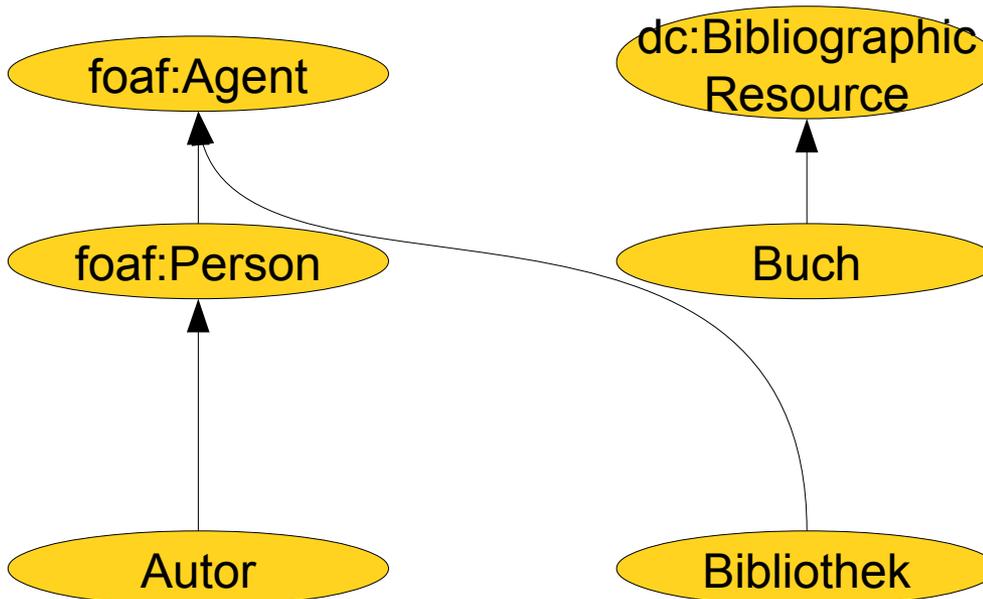
Aufgabe 1

- Klassen anordnen



Aufgabe 1

▪ Klassen anordnen



```
<rdfs:comment
xml:lang="en-US">
A resource that acts or
has the power to act.
</rdfs:comment>
<dcterms:description
xml:lang="en-US">
Examples of Agent
include person,
organization, and
software agent.
</dcterms:description>
```

Aufgabe 1

- Relationen zwischen Klassen

*Eine Bibliothek **besitzt** Bücher. Bibliotheken haben einen Namen, eine Adresse und eine Telefonnummer. Bücher **haben** einen Titel, einen oder mehrere **Autoren**, und eine ISBN-Nummer. Personen haben einen Namen, eine Adresse, eine Telefonnummer und eine E-Mailadresse. Bücher können von einer Person **entliehen** sein.*

Aufgabe 1

- Nachschauen, was in den wiederverwendbaren Schemata schon gibt
 - besitzt: –
 - hat Autor:
 - dc:creator
 - foaf:maker (als äquivalent zu dc:creator definiert)
 - hat ausgeliehen: –

Aufgabe 1

- Literale

*Eine Bibliothek besitzt Bücher. Bibliotheken haben einen **Namen**, eine **Adresse** und eine **Telefonnummer**. Bücher haben einen **Titel**, einen oder mehrere Autoren, und eine **ISBN-Nummer**. Personen haben einen **Namen**, eine **Adresse**, eine **Telefonnummer** und eine **E-Mailadresse**. Bücher können von einer Person entliehen sein.*

Aufgabe 1

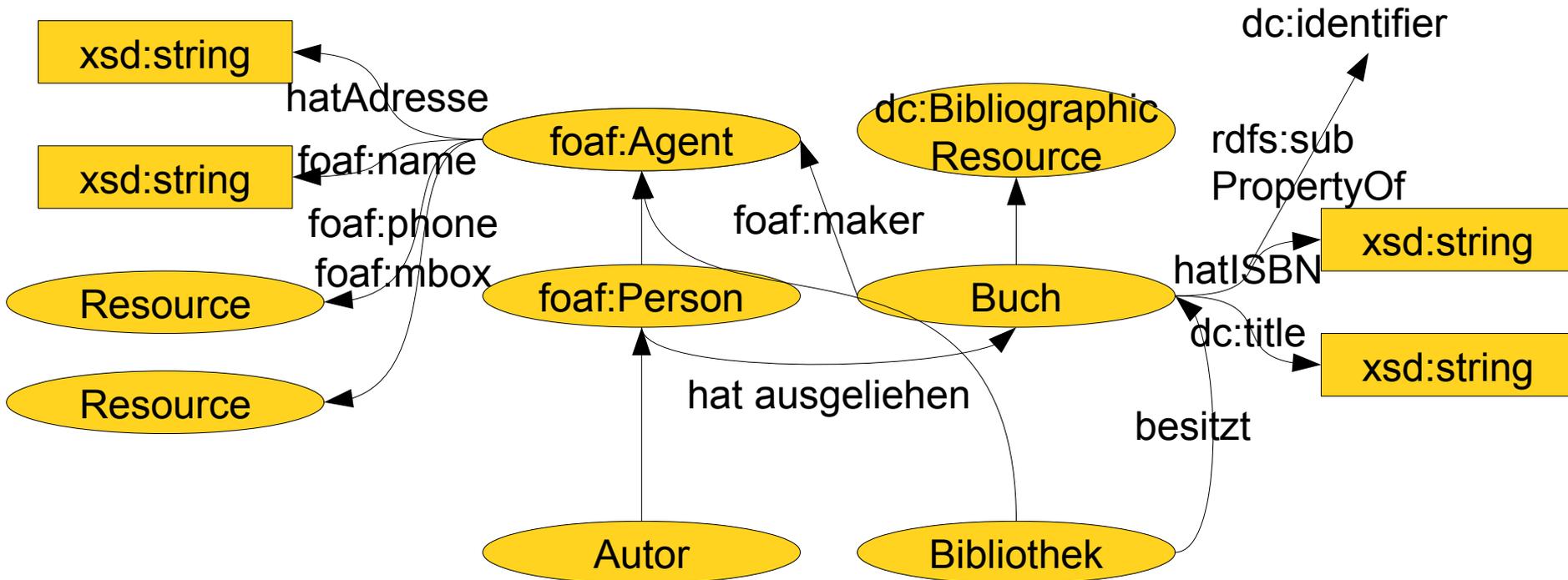
- Nachschauen, was in den wiederverwendbaren Schemata schon gibt
 - Name: z.B. foaf:name
 - Adresse: –
 - Telefonnummer: foaf:phone
 - Titel: dc:title
 - ISBN-Nummer: dc:identifier
 - E-Mail-Adresse: foaf:mbox

Aufgabe 1

- Bibliotheken und Personen haben einige gemeinsame Properties (Name, Adresse, Telefon)
- Was soll die `rdfs:domain` sein?
 - Möglichkeit 1: gemeinsames Superkonzept (`foaf:Agent`)
 - Möglichkeit 2: eigene Properties für jedes definieren
- Warum wäre folgende Definition keine gute Idee?
 - `:hatAdresse rdfs:domain foaf:Person .`
 - `:hatAdresse rdfs:domain :Bibliothek .`
- **RDFS-Reasoning:** aus `:Peter :hatAdresse "Hauptstraße 4"` folgt
 - `:Peter a foaf:Person .`
 - `:Peter a :Bibliothek .`

Aufgabe 1

▪ Fertiges Schema



Aufgabe 1

- Codieren des Schemas
- Wiederverwenden von Klassen und Properties:
 - `:Buch rdfs:subClassOf dc:BibliographicResource .`
 - `:hatISBN rdfs:subPropertyOf dc:identifier .`
 - `:hatAusgeliehen a rdf:Property ;`
`rdfs:domain foaf:Person ;`
`rdfs:range :Buch .`
- Vordefinierte Properties wie `foaf:name` etc. müssen nicht neu definiert werden!

Aufgabe 2

Mit Hilfe des in Aufgabe 1 entwickelten Schemas sollen Informationen über 100 Bibliotheken mit je 1.000.000 Büchern sowie je 10.000 ausleihenden Personen gespeichert werden. Jedes Buch hat im Mittel zwei Autoren, jede Person hat im Mittel fünf Bücher gleichzeitig ausgeliehen. Bestimmen Sie den Speicherbedarf für naive Tripelspeicherung, Property-Table und vertikale Partitionierung.

Hinweis: Die Lösung dieser Aufgabe hängt stark von Ihrer Lösung in Aufgabe 1 ab!

Aufgabe 2



- Naiver Tripel-Store:
 - 100 Bibliotheken mit Typ, Name, Adresse, Telefonnummer:
 $100 * 4$ Tripel
 - $100 * 1.000.000$ Bücher mit Typ, Titel, ISBN, zwei Autoren, Bibliothek:
 $100 * 1.000.000 * 6$ Tripel
 - $100 * 2.000.000$ Autoren mit Typ und Name
 $100 * 2.000.000 * 2$ Tripel
 - $100 * 10.000$ Personen mit Typ, Name, Adresse, Telefon, E-Mail und fünf Bücher:
 $100 * 10.000 * 10$ Tripel
- $400 + 600.000.000 + 400.000.000 + 10.000.000$
 $\sim 1.000.000.000$ Tripel, also 3 Mrd. Strings

Aufgabe 2

- Property Table:
 - 11 Spalten (Subjekt + Typ, Name, Adresse, Telefonnummer, besitzt, Titel, ISBN, hat Autor, hat ausgeliehen, E-Mail)
 - Bibliotheken: besitzen 1.000.000 Bücher, also $100 * 1.000.000$ Zeilen
 - Bücher: haben zwei Autoren, also $100 * 1.000.000 * 2$ Zeilen
 - Autoren: $100 * 1.000.000 * 2$ Zeilen¹
 - Personen: haben fünf Bücher ausgeliehen, also $100 * 10.000 * 5$ Zeilen
- $\sim 500.000.000$ Zeilen à 11 Spalten = 5,5 Mrd. Strings
- Merke: bei jeder Bibliothek bleiben in 999.999 Zeilen 9 Spalten mit NULL!

¹ Wenn man davon ausgeht, dass kein Autor an zwei oder mehr Büchern beteiligt ist

Aufgabe 2

- Vertikale Partitionierung
 - Tabelle rdf:type: $100 + 100 \cdot 1.000.000 + 100 \cdot 10.000$ Einträge
 - Name, Adresse, Telefonnummer: $100 + 100 \cdot 10.000$ Einträge
 - E-Mail: $100 \cdot 10.000$ Einträge
 - Titel: $100 \cdot 1.000.000$ Einträge
 - Autor: $100 \cdot 1.000.000 \cdot 2$ Einträge
 - ISBN: $100 \cdot 1.000.000$ Einträge
 - besitzt: $100 \cdot 1.000.000$ Einträge
 - hatAusgeliehen: $100 \cdot 10.000 \cdot 5$ Einträge
- $\sim 600.000.000$ Einträge à zwei Spalten
- $\sim 1,2$ Mrd. Strings

Aufgabe 3

Auf Folie 53 wurde der einfache RDFS-Reasoning-Algorithmus begonnen. Wie viele Axiome erhalten Sie, wenn Sie den Algorithmus zu Ende ausführen?

Aufgabe 3



```
:Madrid :capitalOf :Spain . (a0)
(a0 + rdf1) :capitalOf rdf:type rdf:Property . (a1)
(a1 + rdf1) rdf:type rdf:type rdf:Property . (a2)
(a0 + rdfs4a) :Madrid rdf:type rdfs:Resource . (a3)
(a0 + rdfs4b) :Spain rdf:type rdfs:Resource . (a4)
(a1 + rdfs4a) :capitalOf rdf:type rdfs:Resource . (a5)
(a1 + rdfs4b) rdf:Property rdf:type rdfs:Resource . (a6)
(a2 + rdfs4a) rdf:type rdf:type rdf:Resource . (a7)
(a3 + rdfs4b) rdf:Resource rdf:type rdf:Resource . (a8)
(a1 + rdfs6) rdf:capitalOf rdfs:subPropertyOf
rdf:capitalOf . (a9)
(a9 + rdf1) rdfs:subPropertyOf rdf:type
rdf:Property . (a10)
(a10+ rdfs4a) rdfs:subPropertyOf rdf:type
rdf:Resource . (a11)
(a2 + rdfs6) rdf:type rdfs:subPropertyOf rdf:type. (a12)
(a10+ rdfs6) rdfs:subPropertyOf rdfs:subPropertyOf
rdfs:subPropertyOf . (a13)
```

Aufgabe 4



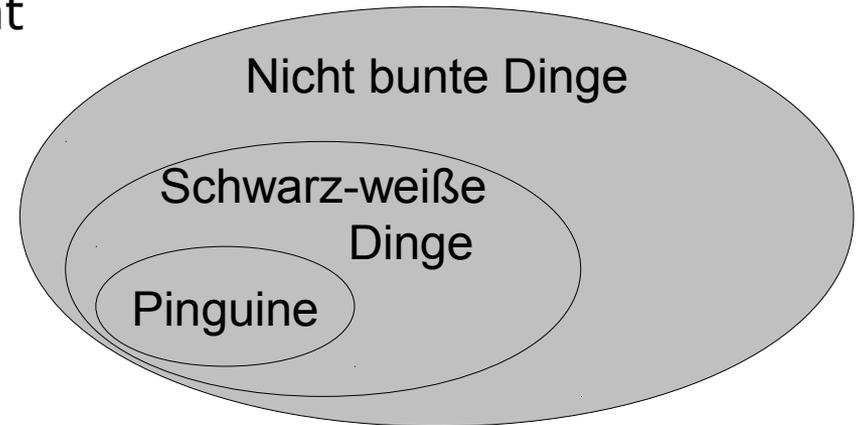
- Was ist der Denkfehler des Pinguins?



Cartoon Copyright: Randy Glasbergen, <http://www.glasbergen.com/>

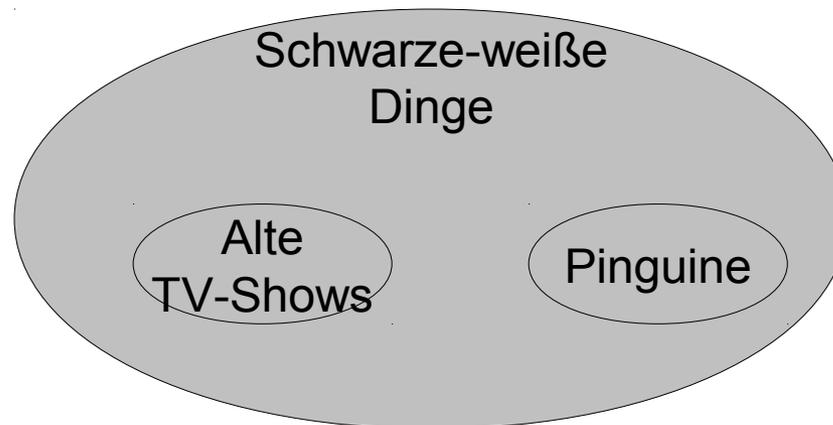
Aufgabe 4

- Wie kann man generell schlussfolgern?
 - Pinguine sind schwarz-weiß.
 - Was schwarz-weiß ist, ist nicht bunt.
 - → Pinguine sind nicht bunt.
- Hier geht man entlang einer Kette:
 - Pinguin → schwarz-weiß → nicht bunt
- ...und findet eine Abkürzung:
 - Pinguin → nicht bunt



Aufgabe 4

- Aber:
 - Pinguin → schwarz-weiß
 - Alte TV-Shows → schwarz-weiß
- Hier gibt es eine solche Kette nicht
 - Und daher ist die Schlussfolgerung auch nicht valide!

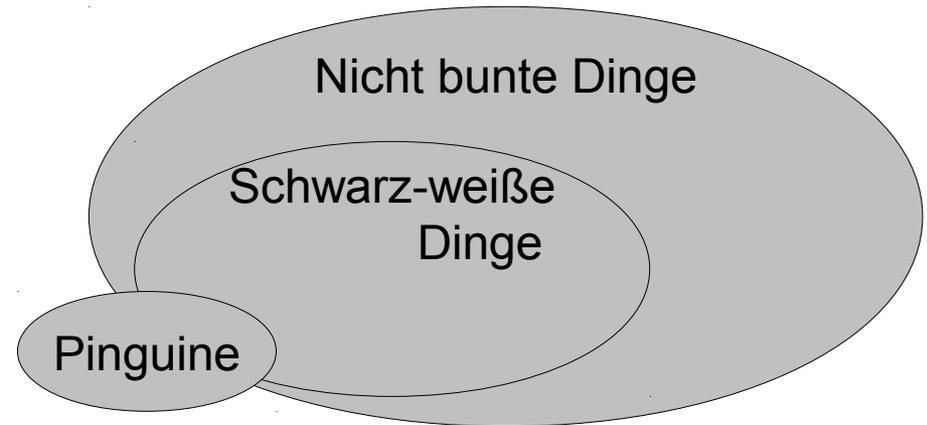


Aufgabe 4

- Exkurs: *some* old TV-shows are black and white
- Das ist ein Existenzquantor:
 - $\exists x: \text{oldTVShow}(x) \rightarrow \text{blackAndWhite}(x)$
- Bei anderen Aussagen: impliziter Allquantor:
 - $\forall x: \text{SchwarzWeiß}(x) \rightarrow \text{nichtBunt}(x)$

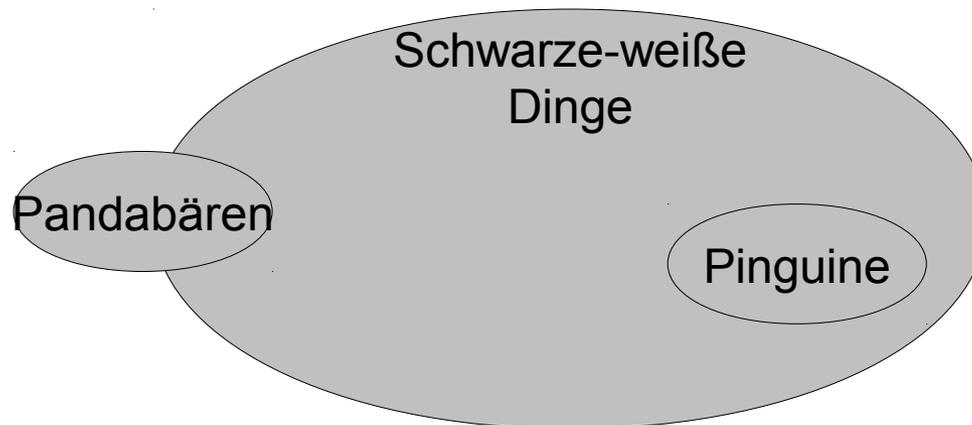
Aufgabe 4

- Den Existenzquantor können wir in solchen Schlussfolgerungen prinzipiell „mitnehmen“, wenn er am Anfang der Kette steht:
 - $\exists x: \text{Pinguin}(x) \rightarrow \text{SchwarzWeiß}(x)$
 - $\forall x: \text{SchwarzWeiß}(x) \rightarrow \text{nichtBunt}(x)$
 - also gilt: $\exists x: \text{Pinguin}(x) \rightarrow \text{nichtBunt}(x)$



Aufgabe 4

- Den Existenzquantor können wir in solchen Schlussfolgerungen *nicht* „mitnehmen“, wenn er in der Mitte der Kette steht:
 - $\forall x: \text{Pinguin}(x) \rightarrow \text{SchwarzWeiß}(x)$
 - $\exists x: \text{SchwarzWeiß}(x) \rightarrow \text{Pandabär}(x)$
 - Daraus folgt nicht, dass manche Pinguine Pandabären sind!



Aufgabe 5

- Gegeben ist folgendes RDF-Schema:
 - :Person a rdfs:Class . (a0)
 - :Student a rdfs:Class . (a1)
 - :Student rdfs:subClassOf :Person . (a2)
 - :University a rdfs:Class . (a3)
 - :enrolledAt a rdf:Property . (a4)
 - :memberOf a rdf:Property . (a5)
 - :memberOf rdfs:domain :Person . (a6)
 - :memberOf rdfs:range :University . (a7)
 - :enrolledAt rdfs:domain :Student . (a8)
- Gegeben ist weiterhin folgende Aussage:
 - :Jana :enrolledAt :TU_Darmstadt . (a9)
- Zeigen Sie, wie ein RDFS-Reasoner daraus die folgende Aussage schließt:
 - :Jana a :Person .

Aufgabe 5



rdfs2 + a9 + a8:
→ :Jana a :Student . (a10)

rdfs9 + a10 + a2:
→ :Jana a :Person . (a11)

| ID | Voraussetzung | Konsequenz |
|-------|--|-----------------|
| rdfs2 | s p o . p rdfs:domain c . | s rdf:type c . |
| rdfs9 | s rdf:type c1 . c1 rdfs:subClassOf c2 . | s rdf:type c2 . |

Aufgabe 6

Peter hat über sich ein FOAF-File veröffentlicht, das folgende Tripel enthält:

@prefix : <http://peter.de/#> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:Peter a foaf:SemanticWebBeginner .

Allerdings gibt es die Klasse "SemanticWebBeginner" nicht in der FOAF-Spezifikation. Gegen welche Regel des Veröffentlichens von Linked Open Data hat Peter damit verstoßen? Schlagen Sie eine Lösung zur Behebung des Problems vor.

Aufgabe 6

- Was passiert, wenn wir diesen URI nachschlagen:
 - foaf:SemanticWebBeginner
 - bzw. `http://xmlns.com/foaf/0.1/SemanticWebBeginner`
- Hier finden wir keine Informationen
 - d.h., dieser URI ist wahrscheinlich nicht auflösbar
- Das sollte aber so sein:
 - *"Verwende auflösbare URIs"*
 - *"Hinterlege an diesen URIs nützliche Informationen, verwende dabei Standards"*
- Gegen diese Prinzipien wurde damit verstoßen

Aufgabe 6

- Wie können wir das beheben?
- Vorschlag 1: vorhandene Klasse nutzen
 - `:Peter a foaf:Person .`
- Vorschlag 2: eigene Klasse definieren
 - `:SemanticWebBeginner a rdfs:Class; rdfs:subClassOf foaf:Person .`
 - `:Peter a :SemanticWebBeginner .`

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2012/2013

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering