Kapitel WT: VIII

VIII. Semantic Web

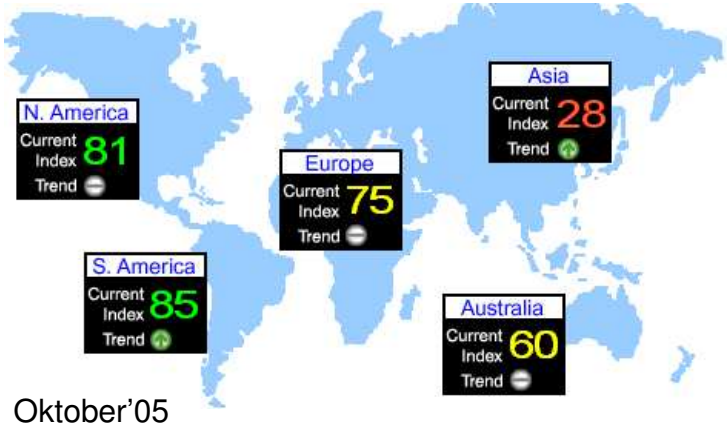
- WWW heute
- Semantic Web Vision
- RDF: Einführung
- RDF: Konzepte
- RDF: XML-Serialisierung
- RDF: Anwendungen
- RDFS: Einführung
- RDFS: Konzepte
- Semantik im Web
- Semantik von RDF/RDFS
- Ontologien
- OWL: Konzepte
- OWL: Logikhintergrund
- OWL: Anwendungen

WWW heute

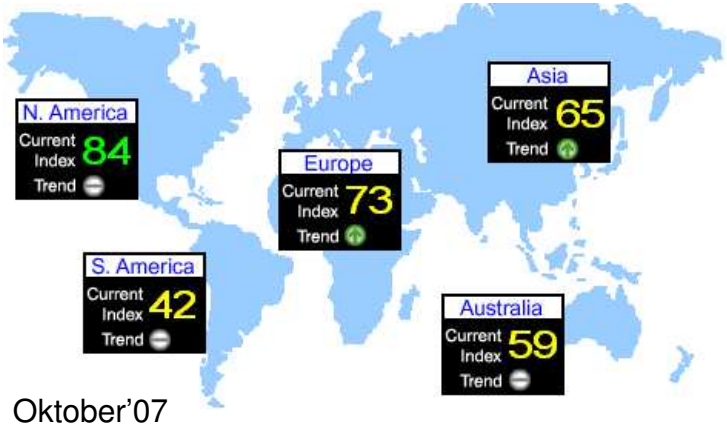
Bestandsaufnahme

Das World Wide Web ist mittlerweile der bedeutendste Dienst auf dem Rechnernetz Internet:

- Antwortzeit, Packet Loss und Up-Time im Internet:



Oktober'05



Oktober'07

[www.internettrafficreport.com]

- Dezember 2004 gab es ca. 50 Milliarden Dokumente im WWW; davon ca. 8 Milliarden bei Google indiziert.
- die Anzahl der Dokumente im WWW verdoppelt sich etwa alle 6 Monate.

Bemerkungen:

- Zur Grafik: *“The Internet Traffic Report monitors the flow of data around the world. It then displays a value between zero and 100. Higher values indicate faster and more reliable connections.”*
- Mehr Informationen zur Statistik, Historie, Benutzung und Benutzer des Internet und des WWW:
  - The Internet Society `www.isoc.org`
  - The Internet Systems Consortium `http://www.isc.org`

WWW heute

Bestandsaufnahme (Fortsetzung)

Das World Wide Web bietet:

- HTTP

  Weltweit kann jeder Rechner Informationsquelle für jeden anderen Rechner sein.

- HTML, XML

  Es steht ein weltweit vereinheitlichtes Datenformat zur Verfügung, das beschreibt, wie Informationen dargestellt werden sollen und verknüpft sind.

- Web-Services

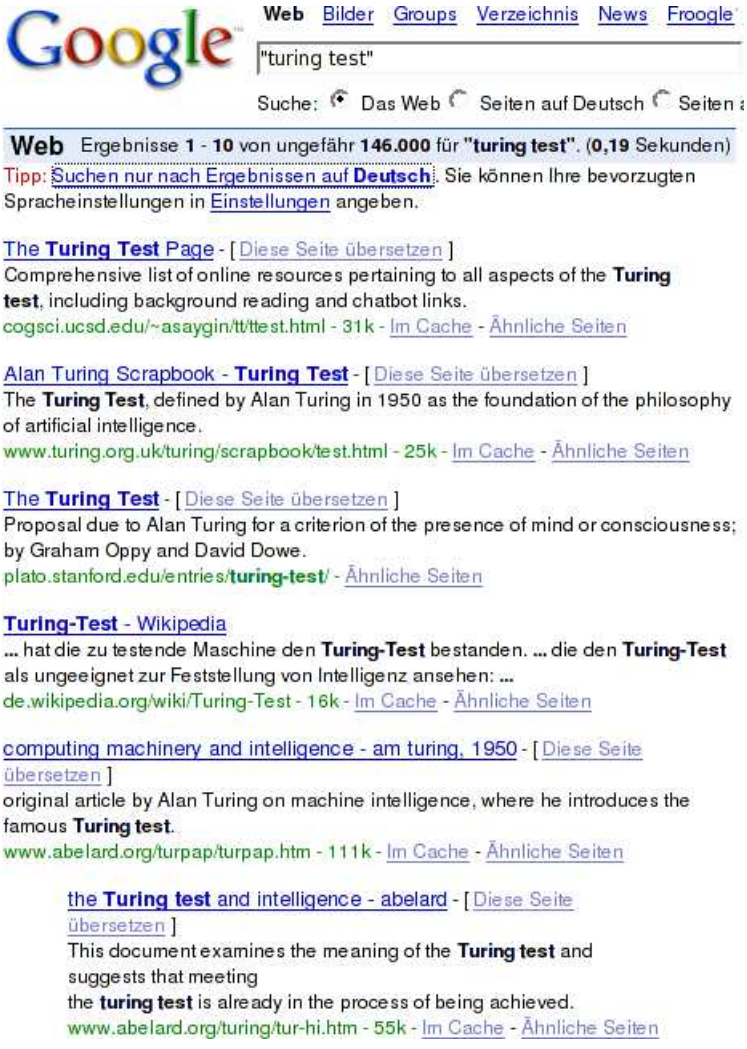
  Weltweit kann jeder Rechner einen Dienst für jeden anderen Rechner zur Verfügung stellen.

Immer mehr:

- Metadaten für die Beschreibung von Semantik.

WWW heute

Herausforderungen [vgl. Sack 2004]



Information Retrieval:

- Synonyme ➜ Recall ↓
- Homonyme ➜ Precision ↓
- keine Kontexteingrenzung
- zuviele oder zuwenige Ergebnisse

Die Rolle von Semantik:

- Begriffsfindung mit Ontologien
- Begriffsabgrenzung mit Ontologien
- Kontextanalyse mit Ontologien
- automatische Anfrageverfeinerung
- personalisiertes Ranking
- dokumentenübergreifende Beantwortung von Anfragen ➜ Textsynthese, Summarization

WWW heute

Herausforderungen

Information Extraction:

- nur von Menschen (gut) durchführbar
- Erkennung von Genre (Texttypen)
- Informationsaggregation aus Bild und Text
- Einschätzung von Informationsqualität

Die Rolle von Semantik:

- Hintergrund- und Weltwissen aus Ontologien
- (einfache) Schlussfolgerungen ziehen

WWW heute

Herausforderungen

Personalisierung:

- Anpassung von Inhalten
- Anpassung der Darstellung
- Erkennung von Verhaltensmustern

Die Rolle von Semantik:

- Verallgemeinerung und Spezialierung von Inhalten (Ontologien)
- Abstimmen mit Gewohnheiten und Präferenzen durch Deduktion

Bemerkungen zu weiteren Herausforderungen:

- Der selbständige Datenaustausch zwischen Maschinen, insbesondere im B2B-Bereich, wächst.
- Information Overload im Web, Informationsbeschaffung zu zeitaufwendig
- Informationsbeschaffung im Web quasi nur durch Menschen durchführbar

Semantic Web Vision

Tim Berners-Lee

“I have a dream for the Web ... and it has two parts.

In the first part, the Web becomes a much more powerful means for collaboration between people. I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse, but to create. [. . .] Furthermore, the dream of people-to-people communication through shared knowledge must be possible for groups of all sizes, interacting electronically with as much ease as they do now in person.

Semantic Web Vision

Tim Berners-Lee

"I have a dream for the Web ... and it has two parts.

In the first part, the Web becomes a much more powerful means for collaboration between people. I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse, but to create. [. . .] Furthermore, the dream of people-to-people communication through shared knowledge must be possible for groups of all sizes, interacting electronically with as much ease as they do now in person.

In the second part of the dream, collaborations extend to computers. Machines become capable of analyzing all the data on the Web—the content, links, and transactions between people and computers.

A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines, leaving humans to provide the inspiration and intuition."

Bemerkungen:

- Noch eine kurze Charakterisierung des Semantic Web:

  "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming."

  [www.w3.org/2001/sw]

- Berners-Lees Idee hinter dem „Semantischen Web" ist die Einsicht, dass es Aufgabe der Informatik ist, dem Menschen bei der Verarbeitung von Daten zu unterstützen. Diese Aufgabe wird durch das heutige Web nur bedingt erfüllt.

  Seine Zielvorstellung ist eine Vereinigung der Eigenschaften einer (Wissens-) Datenbank (Daten durch Maschinen verarbeitbar) mit denen des Webs (jeder kann zu jeder Zeit Daten kreieren und diese beliebig mit anderen verknüpfen), eine Art globale Datenbank.

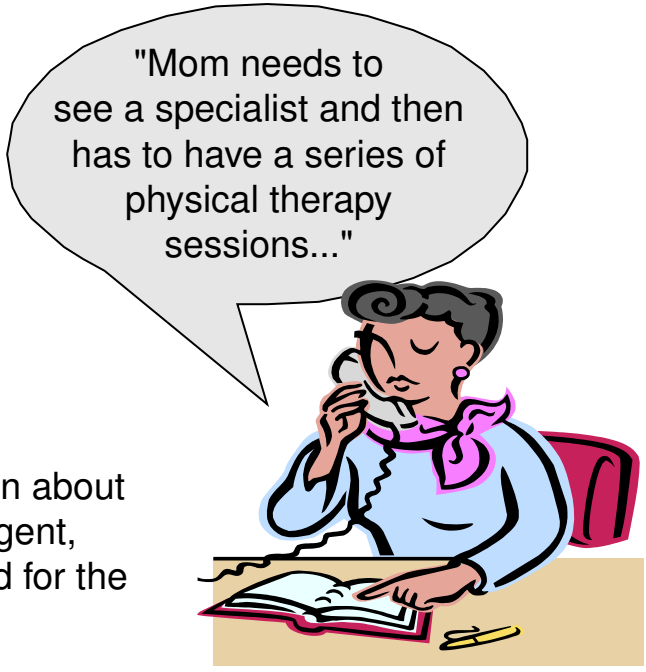
  [Franczyk 2005]

Semantic Web Vision

Beispiel 1: Netzagenten

The phone rang.

When Pete answered, his phone turned the sound down by sending a message to all the other local devices.



Lucy's Semantic Web agent retrieved information about Mom's prescribed treatment from the doctor's agent, looked up several lists of providers, and checked for the ones close to her home ...

... It then began trying to find a match between available appointment times between Pete's and Lucy's busy schedules.

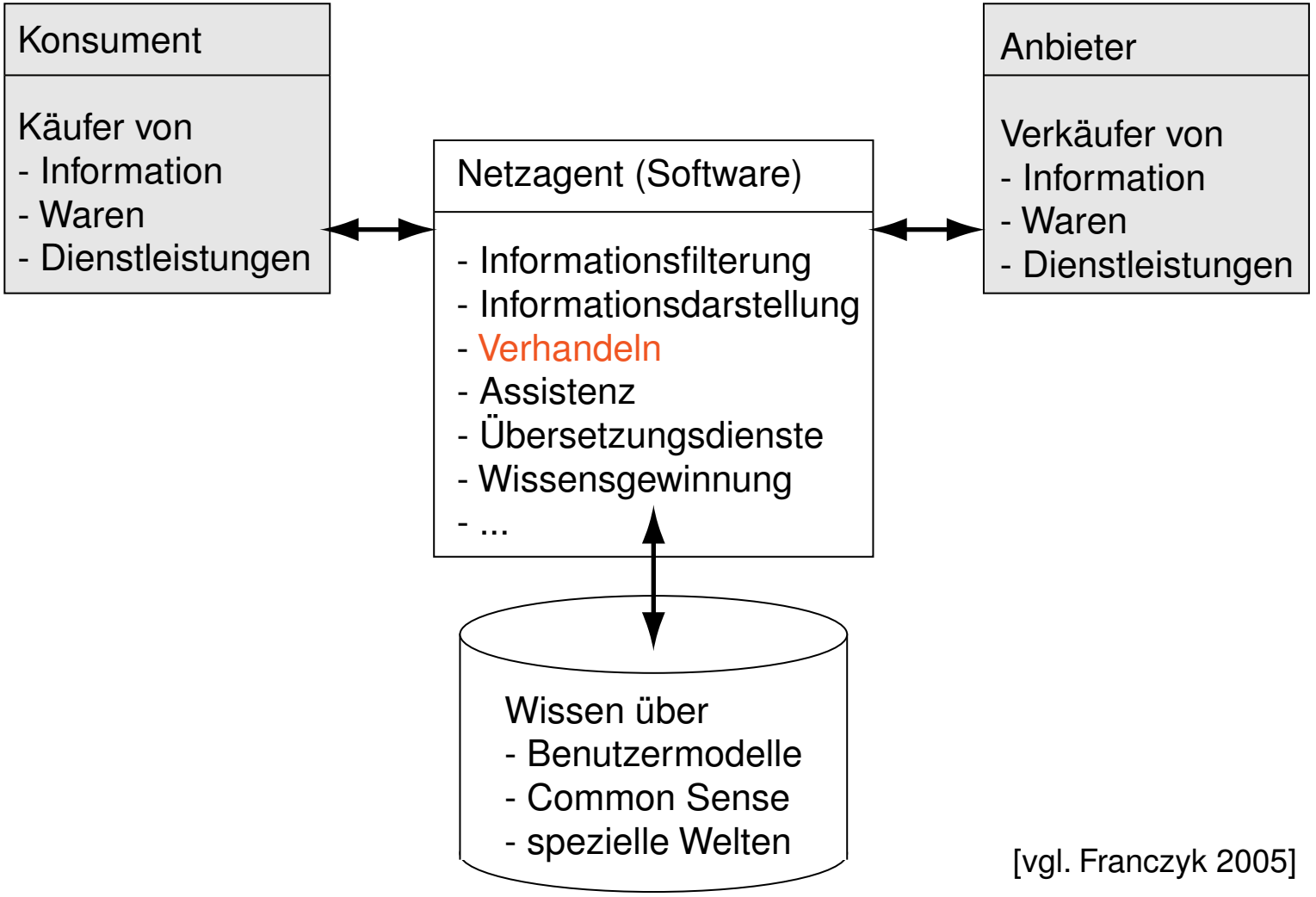
[Berners-Lee 2001]

Bemerkungen:

- Was ist eine Praxis in der Nähe?
- Wie findet man einen guten Arzt?
- Wie geht man mit persönlichen Präferenzen in den Terminkalendern um?

Semantic Web Vision

Beispiel 1: Netzagenten (Fortsetzung)



[vgl. Franczyk 2005]

Semantic Web Vision

Beispiel 2: Deduktive Anfragen

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>
</skill-database>
```

Aufgabe: „Liefere alle Personen.“

Semantic Web Vision

Beispiel 2: Deduktive Anfragen

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>
</skill-database>
```

Aufgabe: „Liefere alle Personen.“

XQuery: `//Person/name`

Ergebnis: `<name>Markus</name>`

Semantic Web Vision

Beispiel 2: Deduktive Anfragen

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>
</skill-database>
```

Aufgabe: „Liefere alle Personen.“

XQuery: `//Person/name`

Ergebnis: `<name>Markus</name>`

Semantic Web Vision

Beispiel 2: Deduktive Anfragen (Fortsetzung)

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>
</skill-database>
```

Aufgabe: „Liefere alle Personen mit SGML-Wissen.“

Semantic Web Vision

Beispiel 2: Deduktive Anfragen (Fortsetzung)

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>

</skill-database>
```

Aufgabe: „Liefere alle Personen mit SGML-Wissen.“

XQuery: `//Person[knowhow=SGML]/name`

Ergebnis: `<name>Markus</name>`

Semantic Web Vision

Beispiel 2: Deduktive Anfragen (Fortsetzung)

```
<skill-database>
<people>
  <Person>
    <name>Markus</name>
    <knowHow>SGML</knowHow>
  </Person>
  <Hacker>
    <name>Jürgen</name>
    <pgp>CB FC A8 17</pgp>
    <knowHow>SGML</knowHow>
    <knowHow>Java</knowHow>
  </Hacker>
  <Person name="Rainer">
    <knowHow>Mike</knowHow>
  </Person>
</people>
```

```
<seminars>
  <Seminar topic="SGML" id="SGM4c">
    <attendant>
      <name>Dieter</name>
      <name>Robert</name>
      <name>Rainer</name>
    </attendant>
  </Seminar>
</seminars>

</skill-database>
```

Aufgabe: „Liefere alle Personen mit SGML-Wissen.“

XQuery: `//Person[knowhow=SGML]/name`

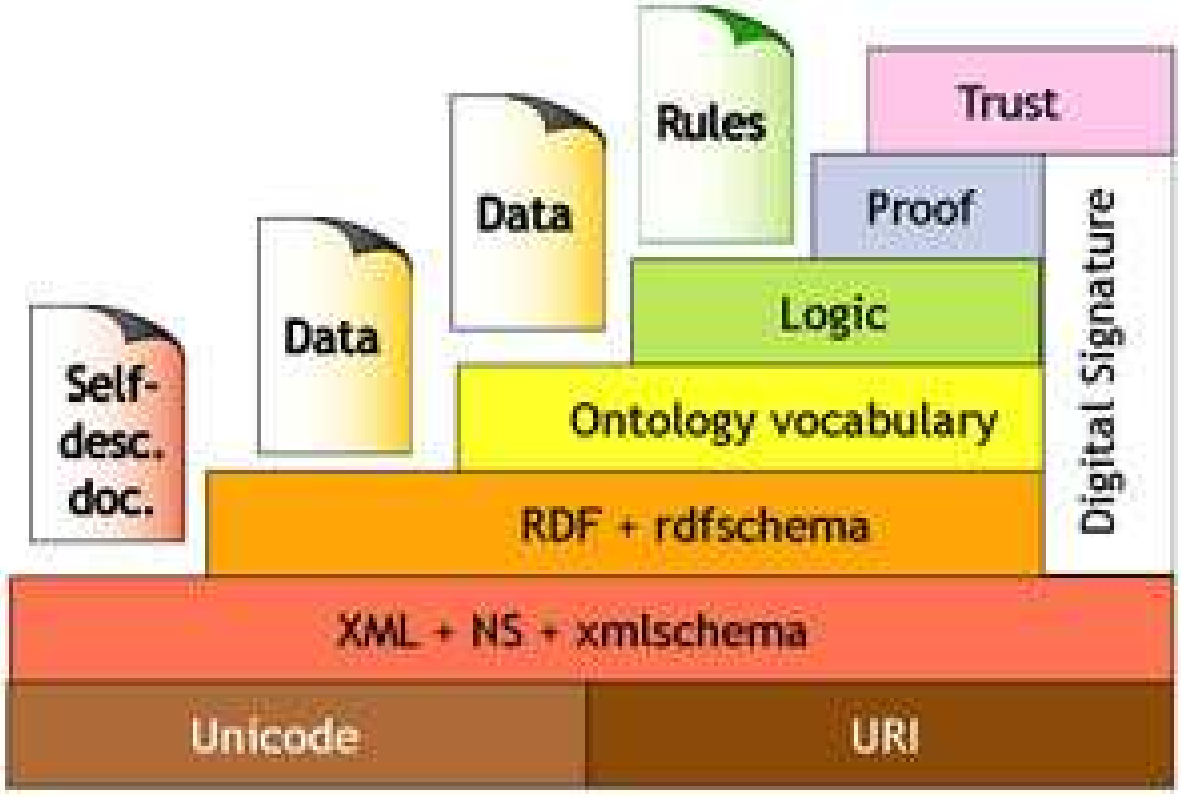
Ergebnis: `<name>Markus</name>`

Bemerkungen:

- Hacker und Seminarteilnehmer sind auch Personen.
- Von der Beschreibung eines Sachverhalts (hier: `name`) kann es syntaktische Varianten geben.
- Common-Sense-Wissen wäre nützlich gewesen: Teilnehmer eines Seminars über SGML werden auch zu Personen, die SGML-Wissen haben.

Semantic Web Vision

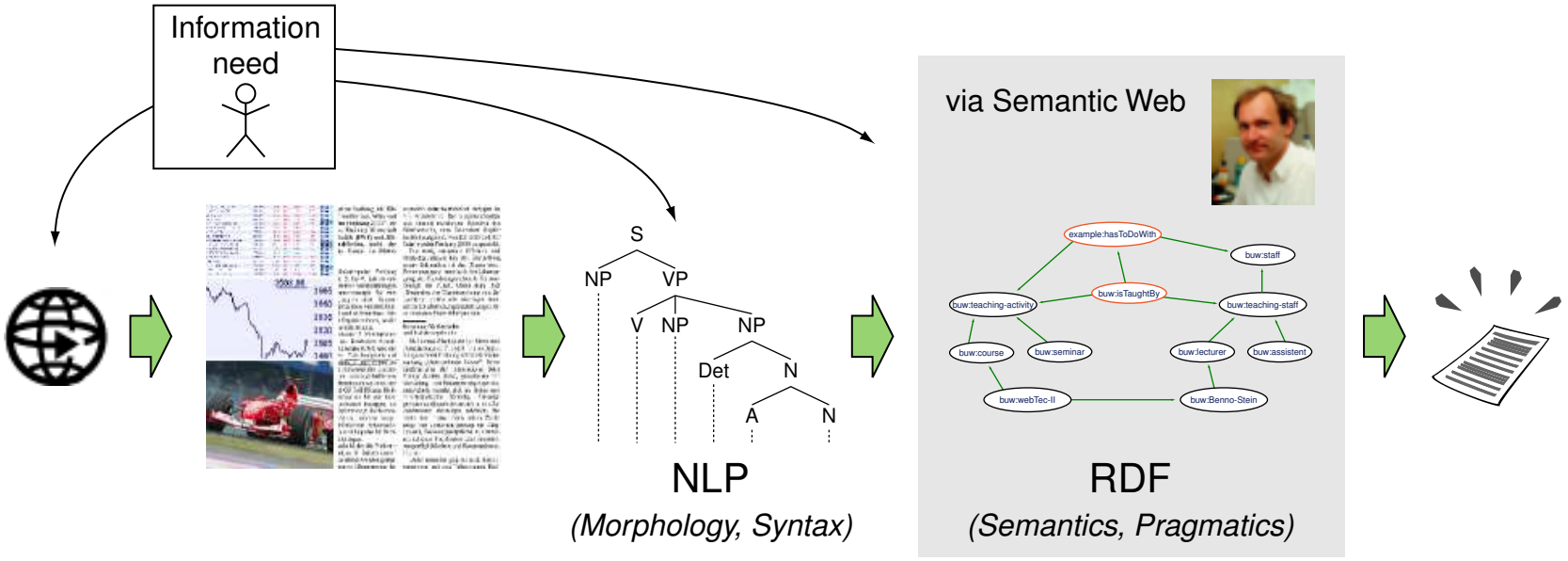
Architektur



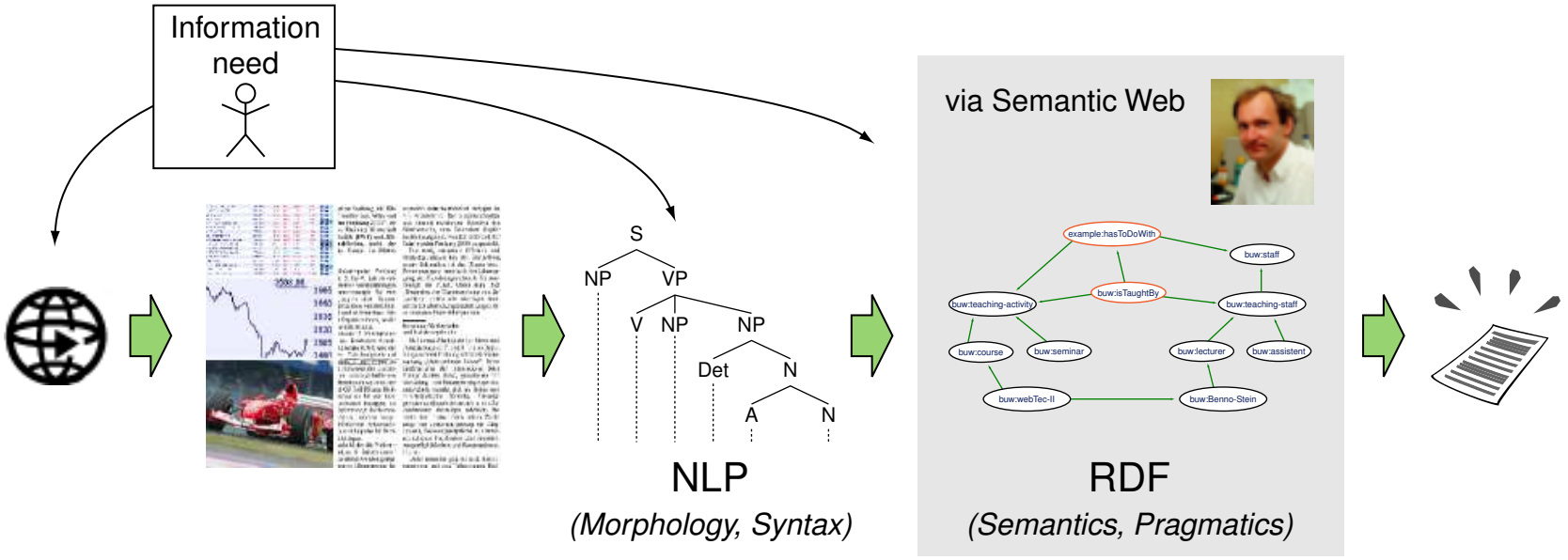
Bemerkungen:

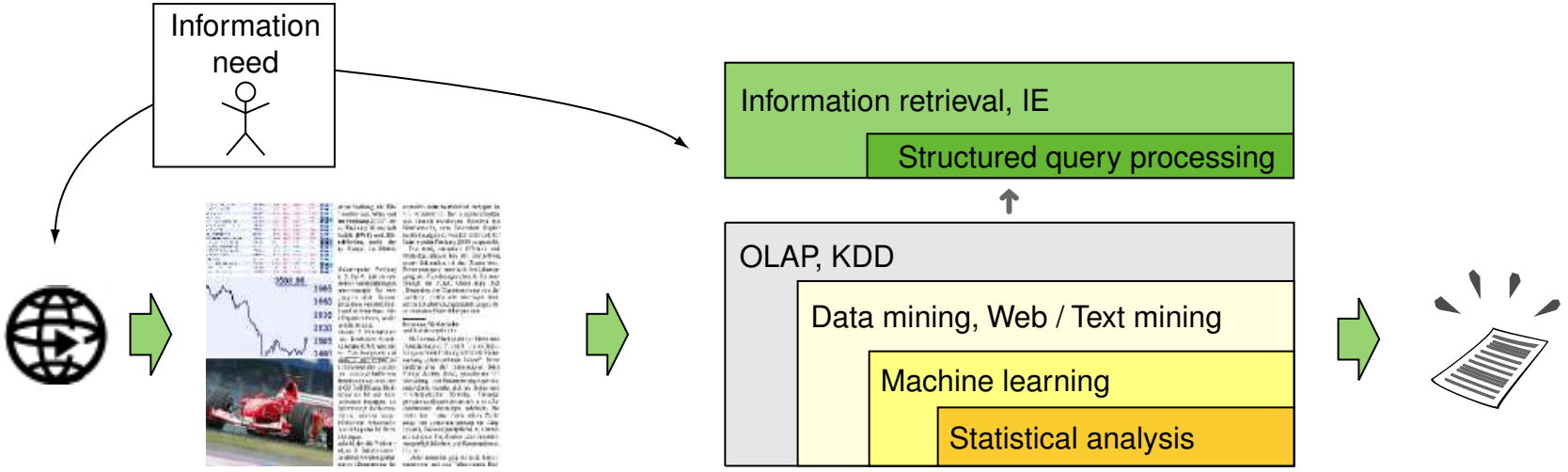
- In den letzten Jahren standen die unteren Schichten im Blickpunkt der Entwicklung und des industriellen Einsatzes. Die Verfechter des Semantic Web hoffen, dass durch die Etablierung der Basistechnologie auch die semantischen Ebenen den Sprung aus der Forschung in die breitere Anwendung schaffen.
- Heute fasst man die Ontologie- und die Logikebene oft in einer Ebene zusammen.

The Semantic Web Way



The Google Way





Semantic Web Vision

Semantic Web Way versus Google Way

Pro Semantic Web. Entwicklung von Semantic Web Technologie:

1998 Berners-Lee. A high-level plan of the architecture of the Semantic WWW.

2004 W3C. OWL—Web Ontology Language. Status: recommendation

Semantic Web Vision

Semantic Web Way versus Google Way

Pro Semantic Web. Entwicklung von Semantic Web Technologie:

1998 Berners-Lee. A high-level plan of the architecture of the Semantic WWW.

2004 W3C. OWL—Web Ontology Language. Status: recommendation

Pro Google. Entwicklung von IR, ML und KDD Technologie:

2002 TAP. Make data available in a machine-understandable format in real time.

2004 IBM. UIMA—The Unstructured Information Management Architecture.

Semantic Web Vision

Semantic Web Way versus Google Way

Pro Semantic Web. Entwicklung von Semantic Web Technologie:

1998 Berners-Lee. A high-level plan of the architecture of the Semantic WWW.

2004 W3C. OWL—Web Ontology Language. Status: recommendation

Pro Google. Entwicklung von IR, ML und KDD Technologie:

2002 TAP. Make data available in a machine-understandable format in real time.

2004 IBM. UIMA—The Unstructured Information Management Architecture.

> *"Unstructured information is the largest and fastest growing source of information available."*
>
> [www.research.ibm.com]

> *"Inferring metadata doesn't work."*
>
> [Tim Bray, Sun, 2005]

Semantic Web Vision

Semantic Web Way versus Google Way

"I'd rather make progress by having computers understand what humans write, than by forcing humans to write in ways computers can understand."

[Sergey Brin, Google co-founder]

Kapitel WT: VIII (Fortsetzung)

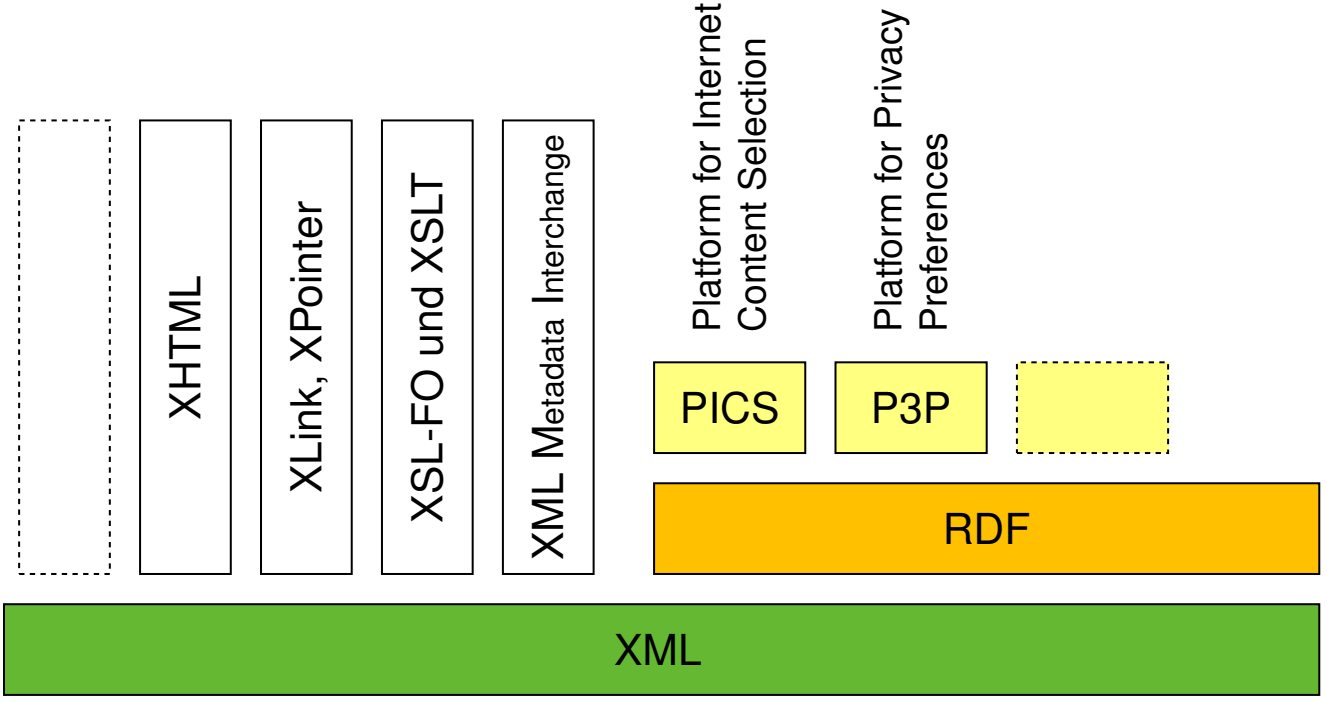
VIII. Semantic Web

- WWW heute
- Semantic Web Vision
- RDF: Einführung
- RDF: Konzepte
- RDF: XML-Serialisierung
- RDF: Anwendungen
- RDFS: Einführung
- RDFS: Konzepte
- Semantik im Web
- Semantik von RDF/RDFS
- Ontologien
- OWL: Konzepte
- OWL: Logikhintergrund
- OWL: Anwendungen

RDF: Einführung

"The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web."

[W3C www.w3.org/TR/rdf-syntax-grammar]



[vgl. Hitzler 2005]

RDF: Einführung

Historie

1998 Berners-Lee. Plan für die Architektur eines Semantic Web.

1999 RDF Schema-Spezifikation. Proposed Recommendation.

2001 RDF/XML Syntax-Spezifikation. Working Draft.

2001 RDF Semantik-Spezifikation. Working Draft.

2002 OWL Web Ontology Language. Working Draft.

2004 RDF/XML Syntax-Spezifikation. Recommendation.

2004 RDF Semantik-Spezifikation. Recommendation.

2004 RDF Vokabularbeschreibungssprache 1.0: RDF-Schema. Recom.

2004 OWL. Web Ontology Language. Semantik und abstrakte Syntax. Recom.

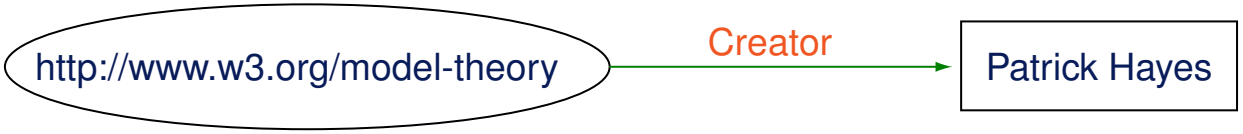
Bemerkungen:

- Die OWL Web Ontology Language hat verschiedene Vorgänger, u. a. die DAML+OIL Ontologie-Sprache, die in der Historie nicht genannt sind. OWL wird als Revision und Weiterentwicklung dieser (zum Teil noch eingesetzten) Ontologie-Sprachen angesehen; wichtiges Ziel ist die Entwicklung eines einheitlichen Standards.

RDF: Einführung

Realistischer Use-Case für das Semantic Web

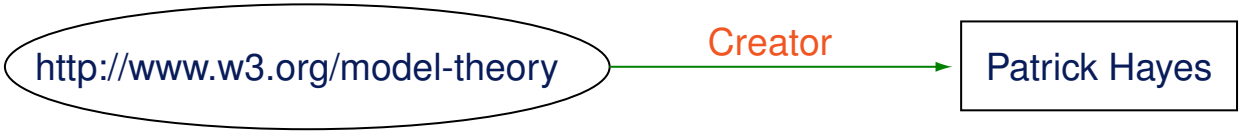
"The creator of resource `http://www.w3.org/model-theory` *is Patrick Hayes."*



RDF: Einführung

Realistischer Use-Case für das Semantic Web

"The creator of resource `http://www.w3.org/model-theory` *is Patrick Hayes."*



Es gibt viele Möglichkeiten, diesen Sachverhalt in XML zu formulieren:

- ```
  <Creator>
    <uri>http://www.w3.org/model-theory</uri>
    <name>Patrick Hayes</name>
  </Creator>
  ```

- ```
  <Document uri="http://www.w3.org/model-theory">
    <Creator>Patrick Hayes</Creator>
  </Document>
  ```

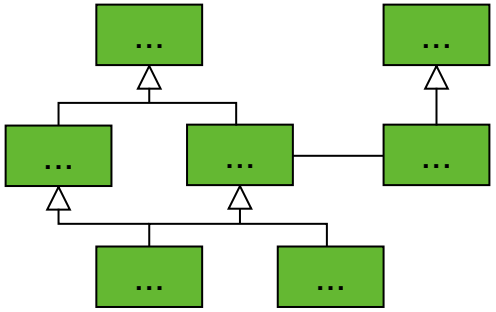
- ```
  <Document uri="http://www.w3.org/model-theory"
      Creator="Patrick Hayes"/>
  ```

➜ Einigung auf das gleiche XML-Schema.
  ```

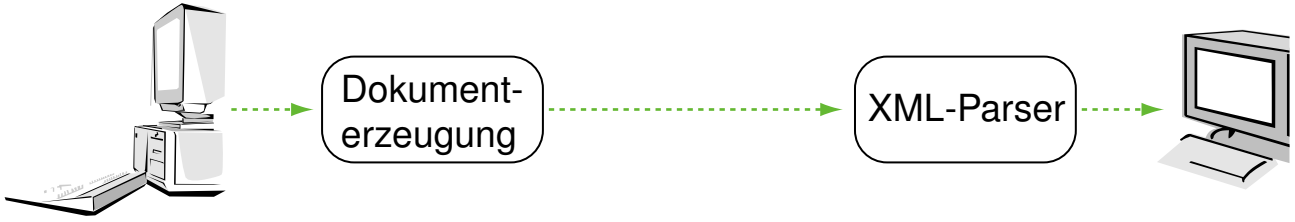
RDF: Einführung

Realistischer Use-Case für das Semantic Web (Fortsetzung)

"Möven sind Vögel.
Vögel sind Wirbeltiere.
Wirbeltiere sind Tiere.
..."



Konzeptuelles
Modell der Domäne

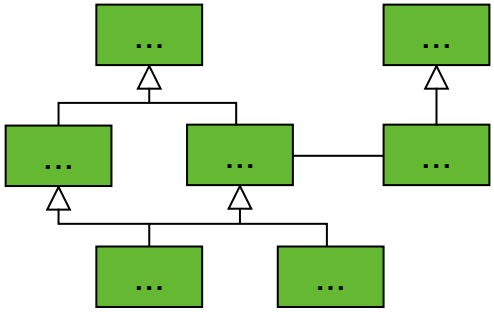


[vgl. Hitzler 2005]

RDF: Einführung

Realistischer Use-Case für das Semantic Web (Fortsetzung)

"Möven sind Vögel.
Vögel sind Wirbeltiere.
Wirbeltiere sind Tiere.
..."



Konzeptuelles
Modell der Domäne

Abbildung

XML-Schema

```
<xsd:schema xmlns:xsd="http://..
      <xsd:annotation> Schema-A
      ...
</xsd:schema>
```

Anwendung

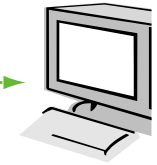
Dokument-
erzeugung

Verwendung
von Schema A

✓

XML-Parser

Verwendung
von Schema A



[vgl. Hitzler 2005]

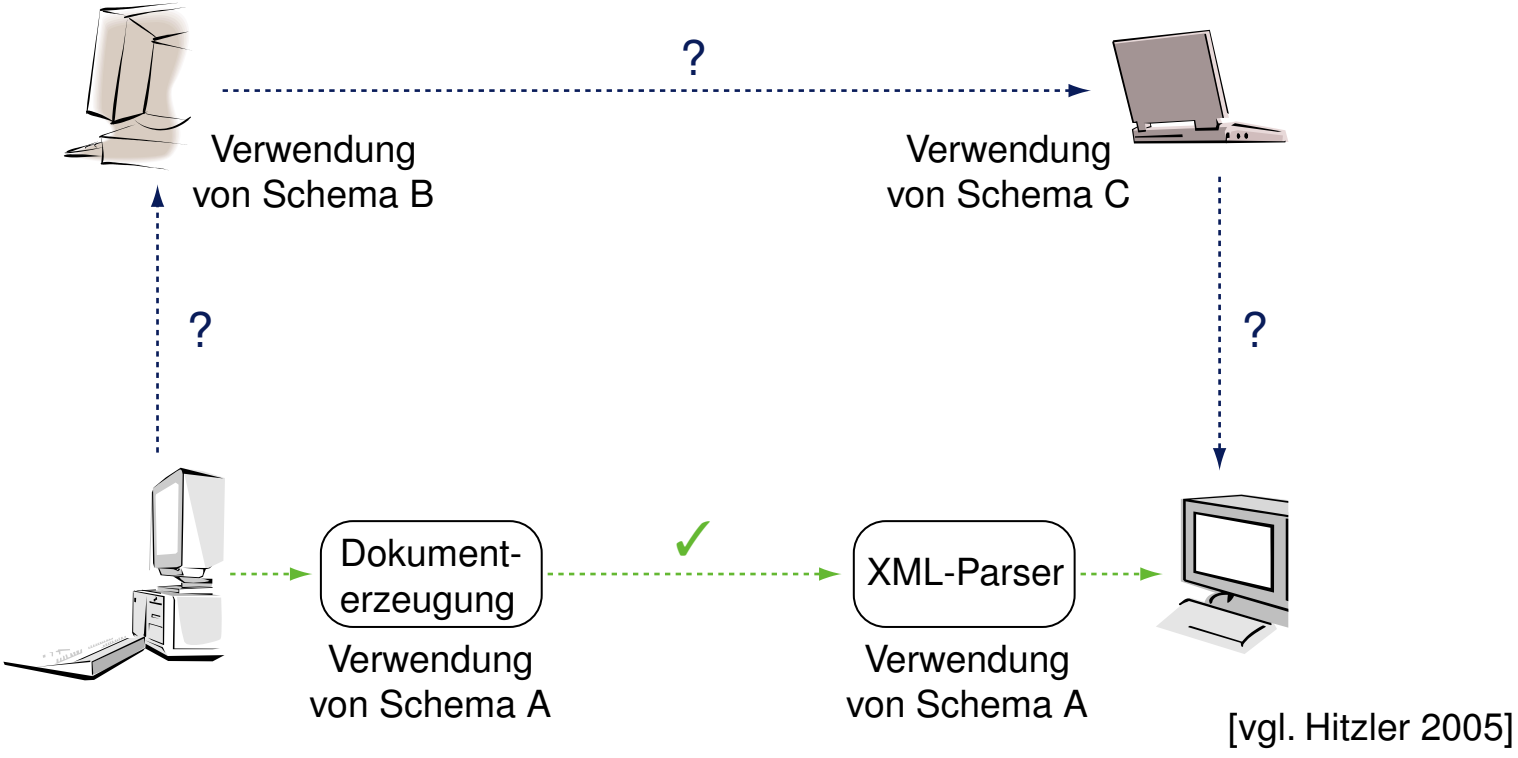
Bemerkungen:

- Sender und Empfänger des Dokumentes haben sich auf eine gemeinsame Semantik verständigt: ihre (XML-basierte) Kommunikation greift auf das gleiche XML-Schema zurück.

RDF: Einführung

Realistischer Use-Case für das Semantic Web (Fortsetzung)

Viele Kommunikationspartner im World Wide Web sind einander zunächst unbekannt und haben sich nicht auf eine gemeinsame Semantik verständigt.

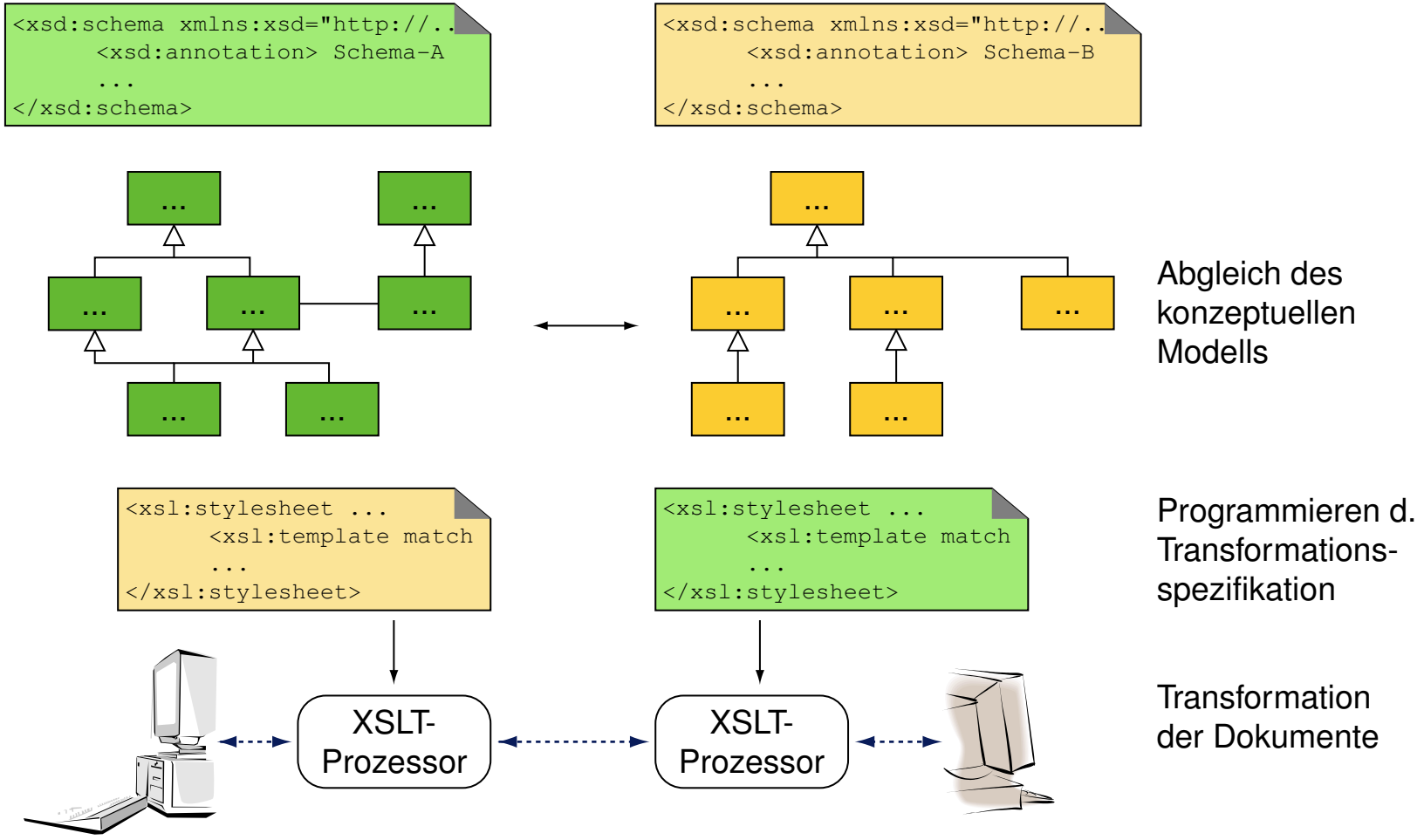


[vgl. Hitzler 2005]

RDF: Einführung

Realistischer Use-Case für das Semantic Web (Fortsetzung)

Notwendige Schritte zur Modell- und Dokumentangleichung:



Bemerkungen:

- Der Abgleich konzeptueller Modelle ist schwierig; die Programmierung der Transformationsspezifikation ist in der Regel kompliziert.
- Ausweg ist die Definition einer weiteren Sprachschicht oberhalb von XML, dem Resource Description Framework, RDF. Auf Basis seiner formalisierten Semantik soll RDF es ermöglichen, notwendige oder sinnvolle Anpassungen, Schematransformationen, etc. automatisch zu erschlussfolgern und durchführen zu können.

RDF: Einführung

Anforderungen an RDF

RDF ist als Basissprachschicht für Wissensrepräsentation im World Wide Web vorgesehen. Anforderungen an eine solche Sprachschicht:

- geeignet, um Daten und Metadaten zu beschreiben
- „Domänen-neutral“ – d.h., es werden keine Annahmen über einen Anwendungsbereich gemacht
- formulierbar (auch) in XML
- einfach und erweiterbar, um möglichst viele Anwendergruppen zu erreichen
- mit Mechanismen, um verteilte Ressourcen im als auch außerhalb des World Wide Web zu spezifizieren
- mit Konzepten zur Beschreibung von Metawissen, um Wissen, Meinungen und Aussagen *über* Wissen zu formulieren

RDF: Einführung

Einordnung von RDF

- XML

  Stellt die Syntax für strukturierte Dokument bereit; keine Formulierung von Constraints bzgl. der Semantik von XML-Dokumenten möglich.

- XML-Schema

  Eine Sprache, um die Struktur von XML-Dokumenten vorzuschreiben; erweitert insbesondere auch die XML-Datentypen.

RDF: Einführung

Einordnung von RDF

- XML

  Stellt die Syntax für strukturierte Dokument bereit; keine Formulierung von Constraints bzgl. der Semantik von XML-Dokumenten möglich.
- XML-Schema

  Eine Sprache, um die Struktur von XML-Dokumenten vorzuschreiben; erweitert insbesondere auch die XML-Datentypen.
- RDF

  Eine universelle Sprache (besser: Datenmodell) zur kanonischen Beschreibung von Ressourcen.
- RDF-Schema

  Zur Beschreibung von Klassenbeziehungen und Eigenschaften für Ressourcen; eine Art einfache Ontologie-Sprache.

RDF: Einführung

Einordnung von RDF

- XML

  Stellt die Syntax für strukturierte Dokument bereit; keine Formulierung von Constraints bzgl. der Semantik von XML-Dokumenten möglich.
- XML-Schema

  Eine Sprache, um die Struktur von XML-Dokumenten vorzuschreiben; erweitert insbesondere auch die XML-Datentypen.
- RDF

  Eine universelle Sprache (besser: Datenmodell) zur kanonischen Beschreibung von Ressourcen.
- RDF-Schema

  Zur Beschreibung von Klassenbeziehungen und Eigenschaften für Ressourcen; eine Art einfache Ontologie-Sprache.
- OWL

  Erweiterte Möglichkeiten, um eigene Terminologien für Ressourcen zu beschreiben: Relationen zwischen Klassen, Kardinalitäts-Constraints, Eigenschaftstypen, aufzählbare Klassen, etc.

Bemerkungen:

- XML-Tags können als eine Art „semantisches Markup“ gesehen werden, wenn für menschliche Leser sinnvolle Bezeichner gewählt sind. Aus *Maschinensicht* ist diese Semantik jedoch nicht gegeben, und der Einsatz von XML soll in erster Linie der maschinellen Generation und Verarbeitung von Dokumenten dienen, z. B. zwischen verschiedenen Firmen und Anwendungen. Ein spezieller und begrenzter Ausweg sind branchenspezifische Festlegungen von gemeinsamen Vokabularen, DTDs und XML-Schemata.
- Die Verwendung von RDF, RDFS oder OWL geschieht durch die Formulierung von RDF-Statements (= SPO-Tripel), deren Elemente aus dem entsprechenden Vokabular (= Menge von URIs) stammen.
- Die Vokabulare, einschließlich Kommentar, Angabe der Signatur bei Prädikat-Ressourcen, Angabe der Oberklasse bei Subjekt-Ressourcen für RDF und RDFS, finden sich in den zugehörigen Namensräumen `http://www.w3.org/1999/02/22-rdf-syntax-ns#` bzw. `http://www.w3.org/2000/01/rdf-schema#`
- Aus Web-Browser-Sicht handelt es sich bei RDF-, RDFS- oder OWL-Code um XML-Code, der lediglich wohlgeformt sein muss.
- Die *Semantik* der aus dem RDF-, RDFS- oder OWL-Vokabular gebauten Sätze (Graphen) ist in den Empfehlungen des W3C auf Basis der Modelltheorie definiert. Die Durchführung von Schlussfolgerungsprozessen und die Operationalisierung der Semantik geschieht in den Anwendungsprogrammen. Hintergründe und grundlegende Darstellung aus modelltheoretischer Sicht: `www.w3.org/TR/rdf-mt`

RDF: Konzepte

Die Schlüsselkonzepte von RDF:

1. graphbasiertes Datenmodell
2. URI-basiertes Vokabular
3. Datentypen
4. Zeichenkonstanten (Literale)
5. Syntax zur Serialisierung in XML
6. Basiskonstrukt zur Formulierung von Aussagen *(Statements)*
7. Inferenz auf Basis der Statements

[W3C www.w3.org/TR/rdf-concepts]

RDF: Konzepte

RDF-Datenmodell

Das RDF-Datenmodell besteht aus drei Objekttypen:

1. Resource
   Eine Ressource ist ein Ding (virtuell oder real), das eine URI hat.
2. Property
   Eine Property ist eine spezielle Art von Ressource, die dazu dient, eine gerichtete Beziehung zwischen einer anderen Ressource R und einem Wert herzustellen. Eine Property kann so zur Beschreibung von R verwendet werden.
3. Statement
   Die konkrete Verwendung einer Property zusammen mit einer Ressource und einem Wert ist ein Statement. Die drei Elemente eines Statements werden *Subjekt*, *Prädikat* und *Objekt* genannt.

Statement = (Thing (Resource) , Property (Resource) , Value (Resource, Literal))

Statement = (Subjekt , Prädikat , Objekt)

Bemerkungen:

- Datenmodelle in der Informatik dienen zur Erfassung und Darstellung der Informations*struktur*, nicht der Information selbst. Datenmodelle definieren die erlaubten Konzepte, um Information zu beschreiben; sie abstrahieren von jeglicher Syntax.

  Beispiel: relationales Datenmodell
- Das Datenmodell von RDF ist sehr einfach; es basiert nur auf den Typen Resource, Property und Statement.
- Das Subjekt und das Prädikat eines Statements müssen eine URI oder ein anonymer Knoten *(Blank node)* sein. Das Objekt kann eine URI, ein anonymer Knoten oder ein Literal sein. Beachte den Unterschied zwischen URI *(Universal Resource Identifier)* und URL *(Universal Resource Locator)*.
- Andere Bezeichnungen für ein Statement sind: RDF-Tripel, Objekt-Attribut-Wert-Tripel, OAW- bzw. OAV-Tripel, SPO-Tripel
- Das Datenmodell zeigt eine enge Verwandtschaft zu den Frame-Systemen und den semantischen Netzen aus dem Bereich der Künstlichen Intelligenz.

RDF: Konzepte

RDF-Datenmodell

Statement:

(a) *"The creator of resource* `http://www.w3.org/model-theory` *is Patrick Hayes."*

(b) *"Resource* `http://www.w3.org/model-theory` *has the creator Patrick Hayes."*

- Resource bzw. Subjekt: `http://www.w3.org/model-theory`
- Property bzw. Prädikat: `http://www.w3.org/#Creator`
- Wert bzw. Objekt: „Patrick Hayes"

RDF: Konzepte

RDF-Datenmodell

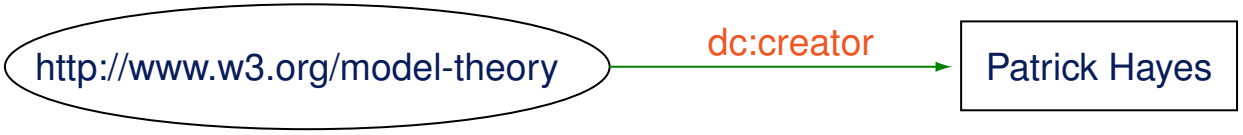
Statement:

(a) *"The creator of resource* `http://www.w3.org/model-theory` *is Patrick Hayes."*

(b) *"Resource* `http://www.w3.org/model-theory` *has the creator Patrick Hayes."*

- Resource bzw. Subjekt: `http://www.w3.org/model-theory`
- Property bzw. Prädikat: `http://www.w3.org/#Creator`
- Wert bzw. Objekt: „Patrick Hayes"

Darstellung als gerichteter Graph:



Darstellung als Prädikat:

Creator(`http://www.w3.org/model-theory`, „Patrick Hayes")

Bemerkungen:

- Bei Aussage (b) ist das RDF-Subjekt auch das grammatikalische Subjekt des Satzes; bei Aussage (a) ist das grammatikalische Subjekt *"The creator"*. Beachte die Rolle von Agens und Patiens (siehe Lexikon unter `culturitalia.uibk.ac.at/hispanoteca`).
- Dinge, die URIs besitzen (Ressourcen), werden durch Ellipsen dargestellt; Literale werden durch Kästchen dargestellt.
- In RDF ist nur die Definition binärer Prädikate möglich.
- In RDF wird eine Menge von URIs (innerhalb eines Graphen, innerhalb einer Anwendung) auch als Vokabular *(Vocabulary)* bezeichnet.

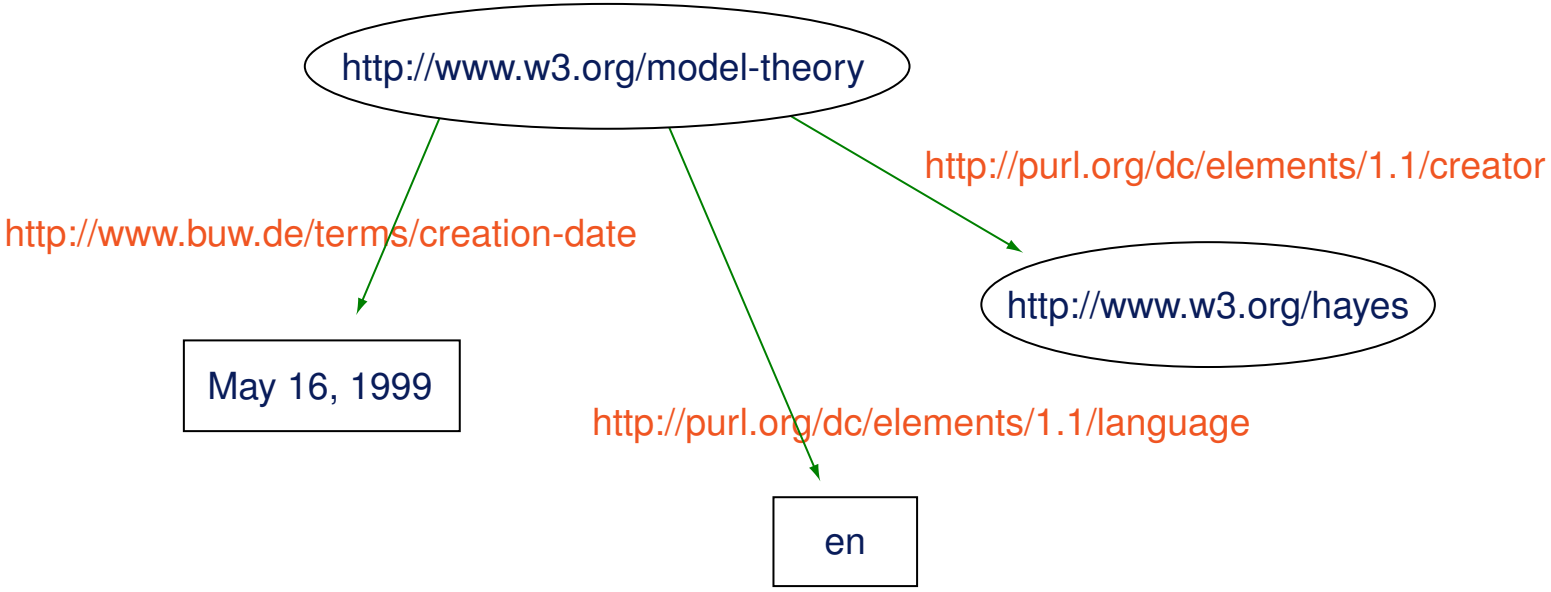
RDF: Konzepte

Semantisches Netz

Gleiche Elemente in Mengen von Statements werden unifiziert und bilden einen eventuell zusammenhängenden Graph.

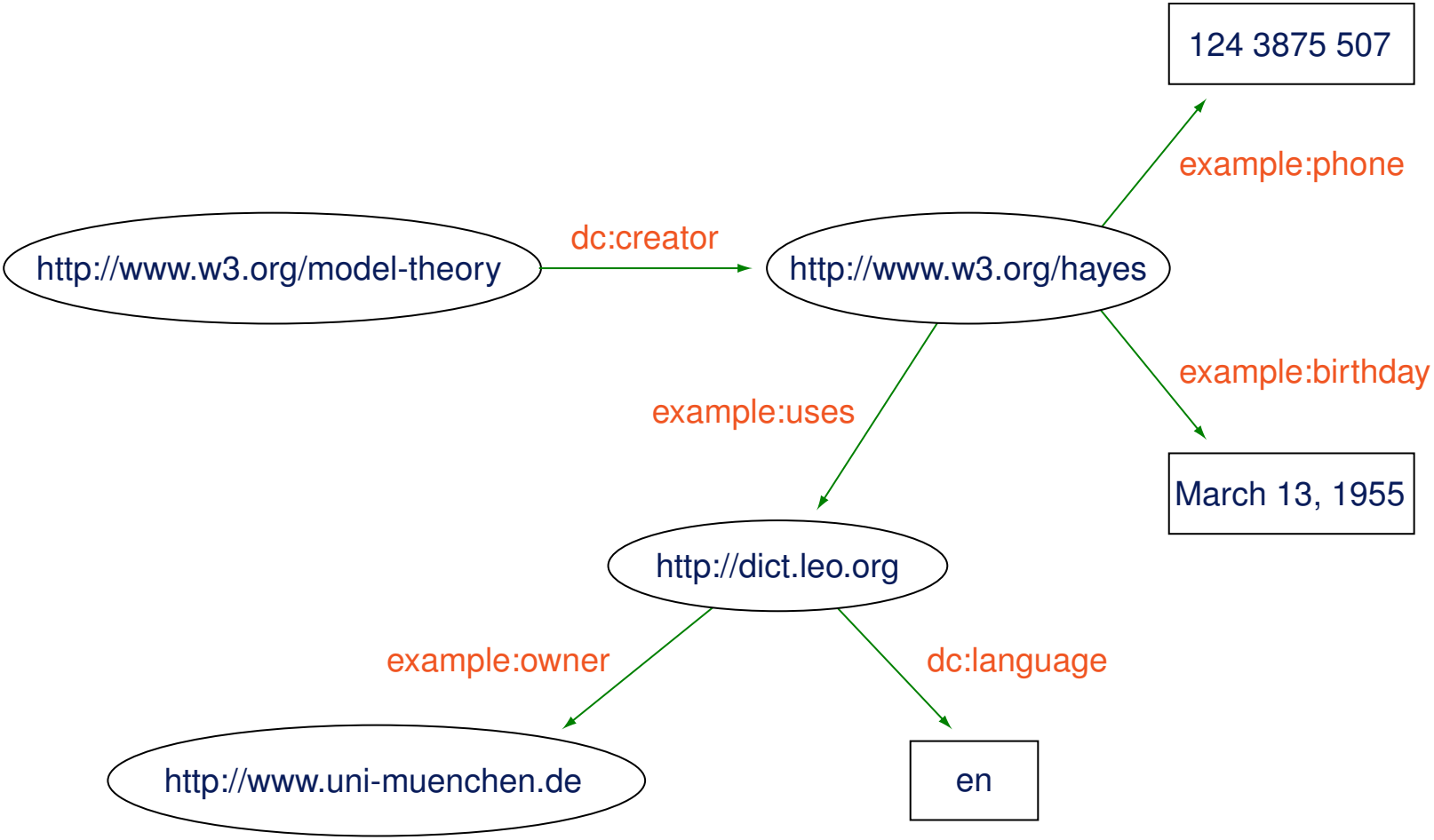
Statements:

"The creator of `http://www.w3.org/model-theory` *is* `http://www.w3.org/hayes`.
`http://www.w3.org/model-theory` *has a creation-date whose value is May 16, 1999.*
`http://www.w3.org/model-theory` *has a language whose value is English."*



RDF: Konzepte

Semantisches Netz

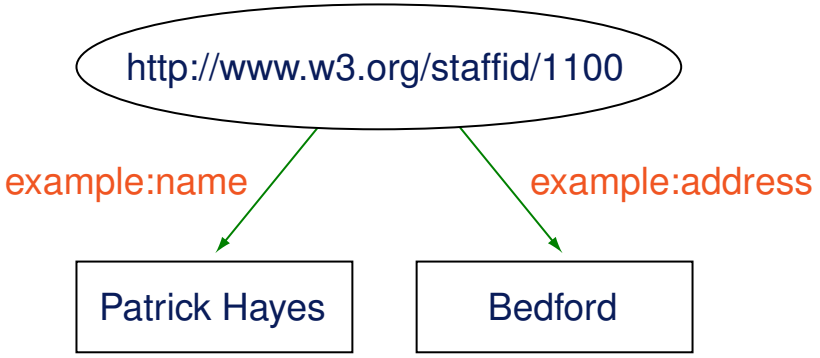


Bemerkungen:

- Der Prefix `dc:` steht für die Namensraum-URI `http://purl.org/dc/elements/1.1/`.
- Der Prefix `example:` steht für die Namensraum-URI `http://www.buw.de/example/`.

RDF: Konzepte

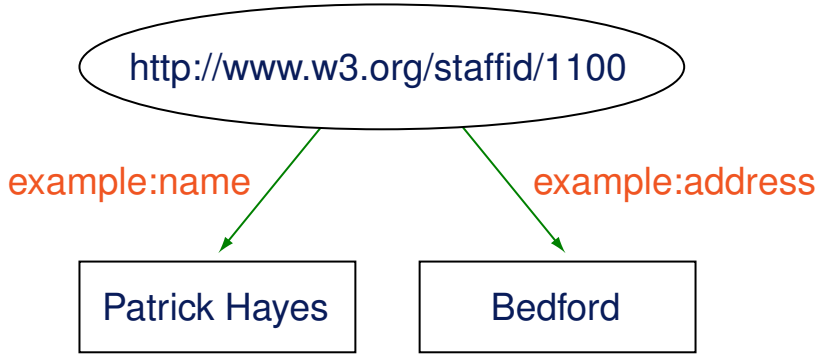
Binarization



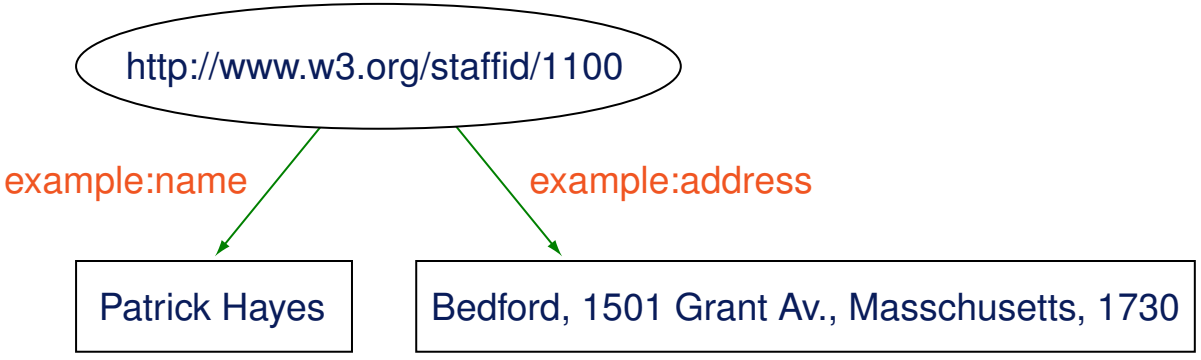
Als logische Formel: $\alpha = \mathit{Name}(1100, \text{"Hayes"}) \wedge \mathit{Address}(1100, \text{"Bedford"})$

RDF: Konzepte

Binarization



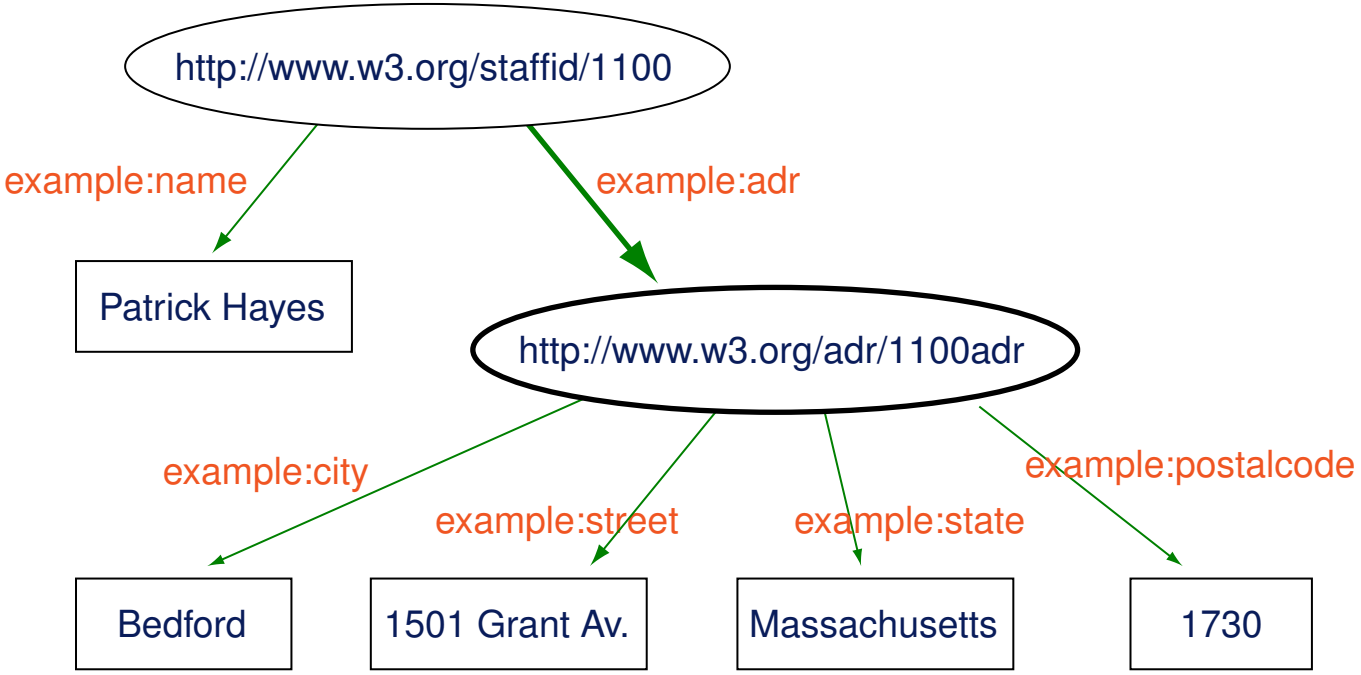
Als logische Formel: $\alpha = \mathit{Name}(1100, \text{"Hayes"}) \wedge \mathit{Address}(1100, \text{"Bedford"})$



$\beta = \mathit{Name}(1100, \text{"Hayes"}) \wedge$
$\mathit{Address}(1100, \text{"Bedford"}, \text{"1501 Grant Av."}, \text{"Massachusetts"}, 1730)$

RDF: Konzepte

Binarization



$\gamma =$ *Name*(1100, "Hayes") $\wedge$ *Adr*(1100, 1100adr) $\wedge$
City(1100adr, "Bedford") $\wedge$ *Street*(1100adr, "1501 Grant Av.") $\wedge$
State(1100adr, "Massachusetts") $\wedge$ *PostalCode*(1100adr, 1730)

β und γ sind logisch äquivalent: $\beta \approx \gamma$

Bemerkungen:

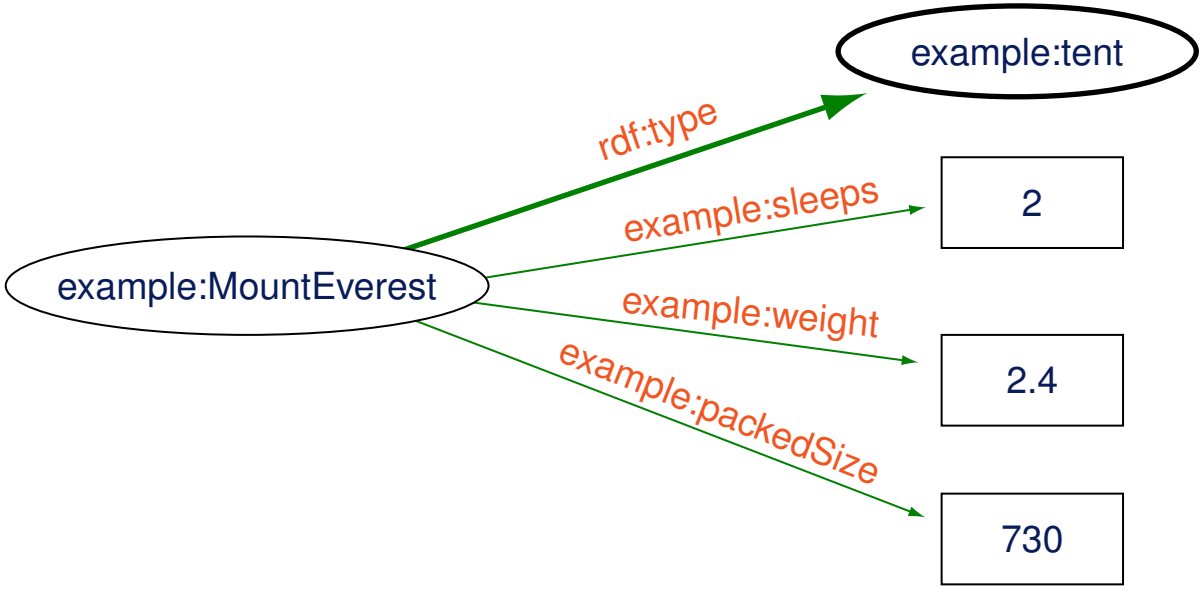
- Die Reformulierung n-stelliger Prädikate als zweistellige Prädikate (Binarization) entspricht der Transformation eines Hypergraphen in einen ordinären Graphen. In einem Hypergraph kann eine Kante mehr als zwei Knoten verbinden.
- Beachte auch die Verwandtschaft zur Normalformbildung in relationalen Datenbankschemata.
- Binarization erfordert die Generierung eines neuen Knoten bzw. einer neuen Ressource. Diese Knoten dürfen anonym, also unbenannt bleiben; in RDF-Terminologie werden anonyme Knoten als „Blank Nodes“ bezeichnet.

RDF: Konzepte

Getypte Ressourcen

Mit der Property `rdf:type` wird ein Statement zu Angabe einer *Klasse* gemacht, zu denen eine Ressource gehört.

- Der Wert (bzw. das Objekt) des Statements ist eine Web-Ressource, die eine Klasse repräsentiert.
- Zu einer Ressource sind mehrere `rdf:type`-Statements möglich.



Bemerkungen:

- Der Prefix `rdf:` steht für die Namensraum-URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.
- Die Klassenangabe `rdf:type` besitzt nur deklarativen Charakter innerhalb eines RDF-Modells. Es ist Aufgabe der Anwendung, die ein RDF-Modell verarbeitet, die intendierte Semantik zu operationalisieren.

RDF: Konzepte

Container

Container in RDF dienen dazu, eine Menge von Ressourcen oder Literalen in ihrer Gesamtheit zu behandeln. Für drei Container-Typen sind entsprechende Terme im RDF-Vokabular definiert:

- `rdf:Bag`
  Ungeordnete Liste von Ressourcen oder Literalen mit Mehrfachvorkommen.
- `rdf:Seq`
  Geordnete Liste von Ressourcen oder Literalen mit Mehrfachvorkommen.
- `rdf:Alt`
  Menge von alternativen Ressourcen oder Literalen ohne Mehrfachvorkommen.

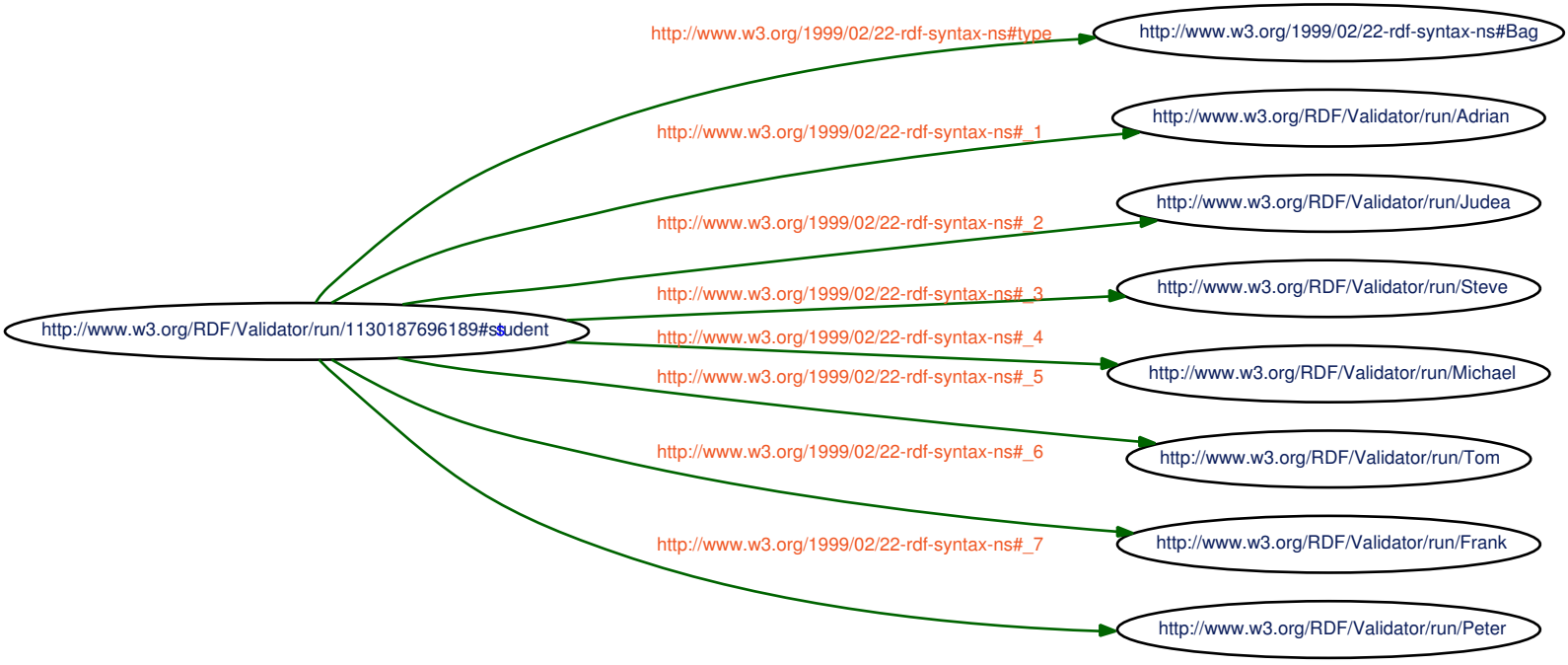
Container stellen getypte Ressourcen dar, deren weitere Eigenschaften generische Properties sind. Die Werte der generischen Properties sind die Inhalte des Containers.

RDF: Konzepte

Container [Serialization]

Beispiel:

„Die Studenten (einer Vorlesung) sind Adrian, Judea, Steve, Michael, Tom, Frank, Peter.“



Bemerkungen:

- Die Bezeichner der generischen Properties sind entweder `rdf:_1`, `rdf:_2`, etc. oder alternativ `rdf:li`.
- Die drei Containertypen `rdf:Bag`, `rdf:Seq`, `rdf:Alt` besitzen nur deklarativen Charakter innerhalb eines RDF-Modells. Es ist Aufgabe der Anwendung, die ein RDF-Modell verarbeitet, die intendierte Semantik zu operationalisieren:

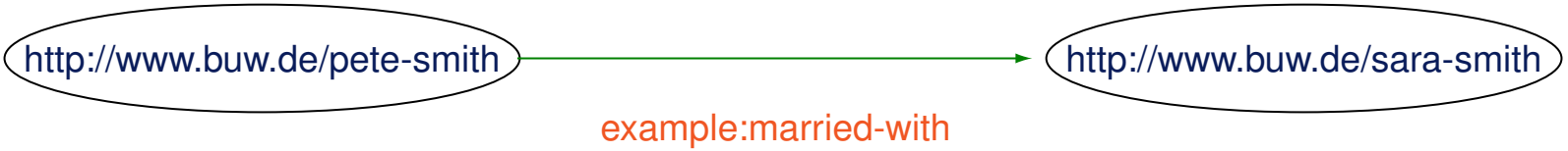
  "It is important to understand that while these types of containers are described using predefined RDF types and properties, any special meanings associated with these containers, e.g., that the members of an Alt container are alternative values, are only intended meanings. These specific container types, and their definitions, are provided with the aim of establishing a shared convention among those who need to describe groups of things."

  [W3C www.w3.org/TR/rdf-primer]
- Der Graph wurde mit dem W3C RDF-Validator automatisch aus einer XML-Serialisierung des RDF-Modells erzeugt. Siehe www.w3.org/RDF/Validator

RDF: Konzepte

Reification [Serialization]

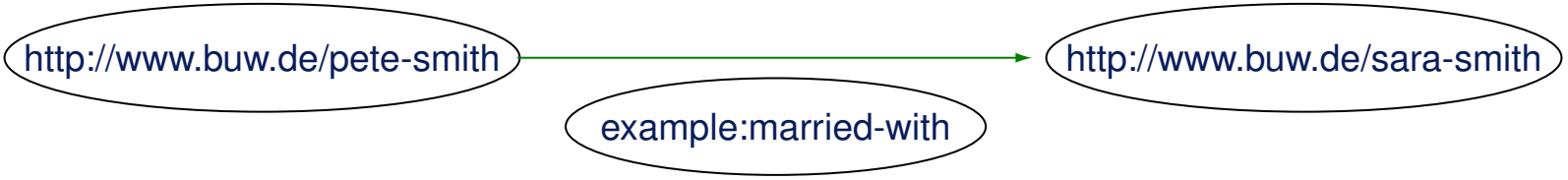
Idee: Für ein Statement wird eine neue Ressource erstellt, die wiederum in einem anderen Statement verwendet werden kann.



RDF: Konzepte

Reification [Serialization]

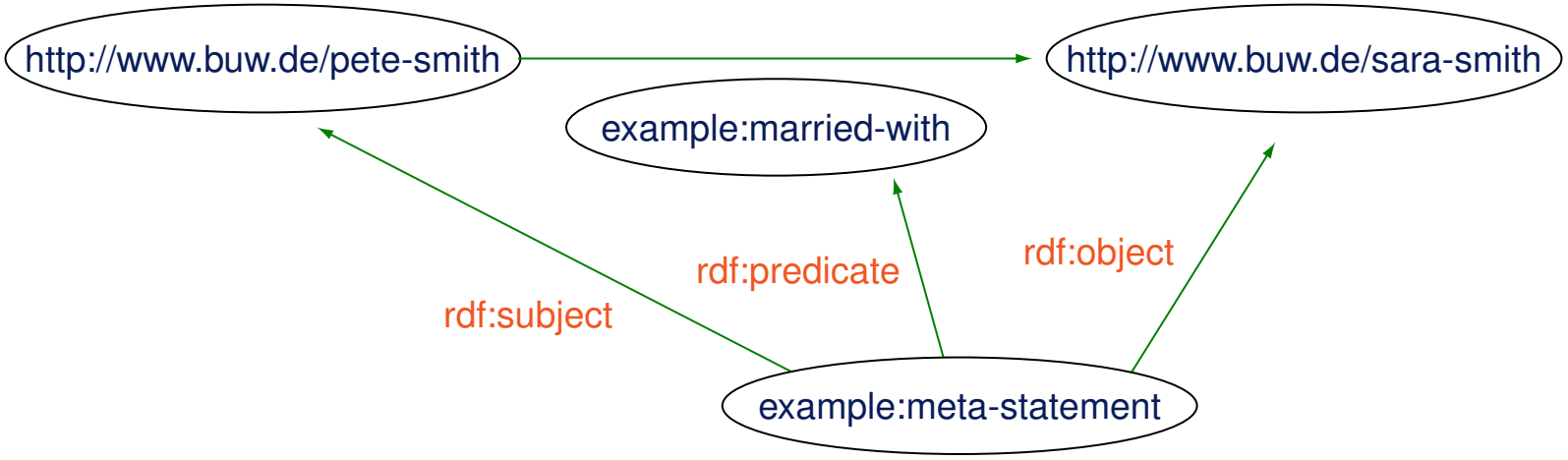
Idee: Für ein Statement wird eine neue Ressource erstellt, die wiederum in einem anderen Statement verwendet werden kann.



RDF: Konzepte

Reification [Serialization]

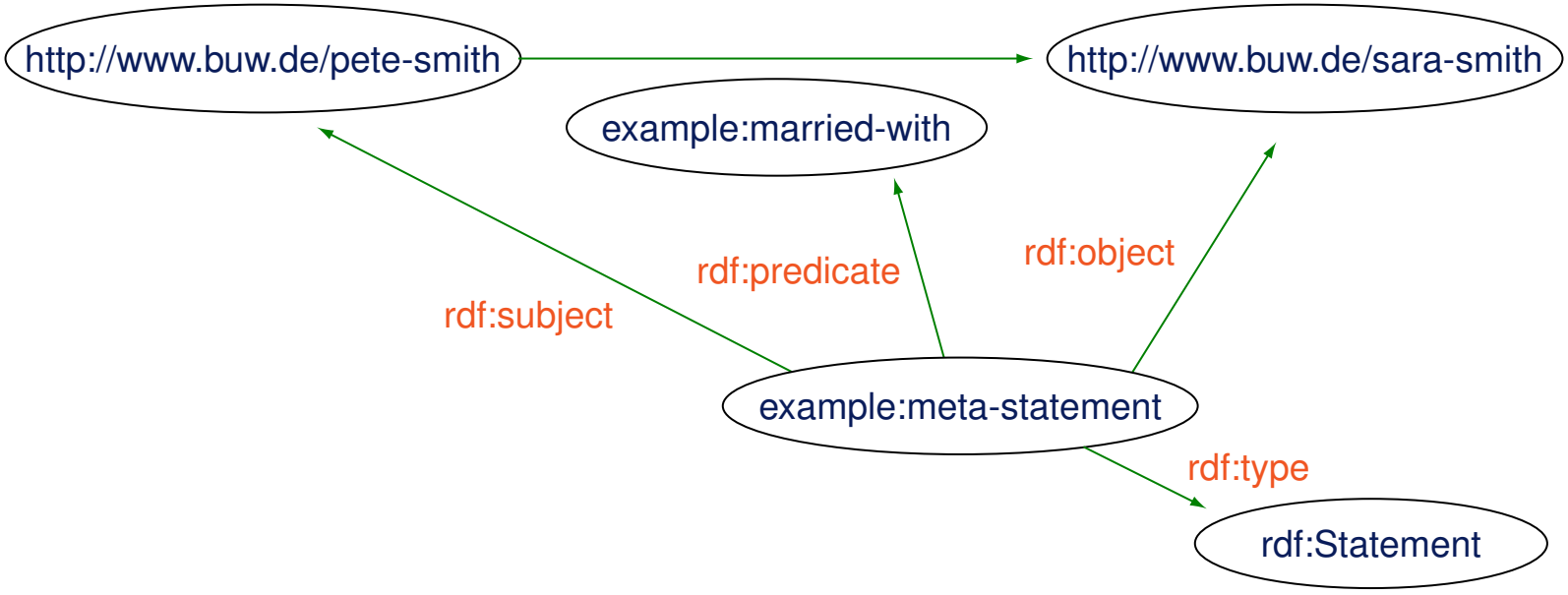
Idee: Für ein Statement wird eine neue Ressource erstellt, die wiederum in einem anderen Statement verwendet werden kann.



RDF: Konzepte

Reification [Serialization]

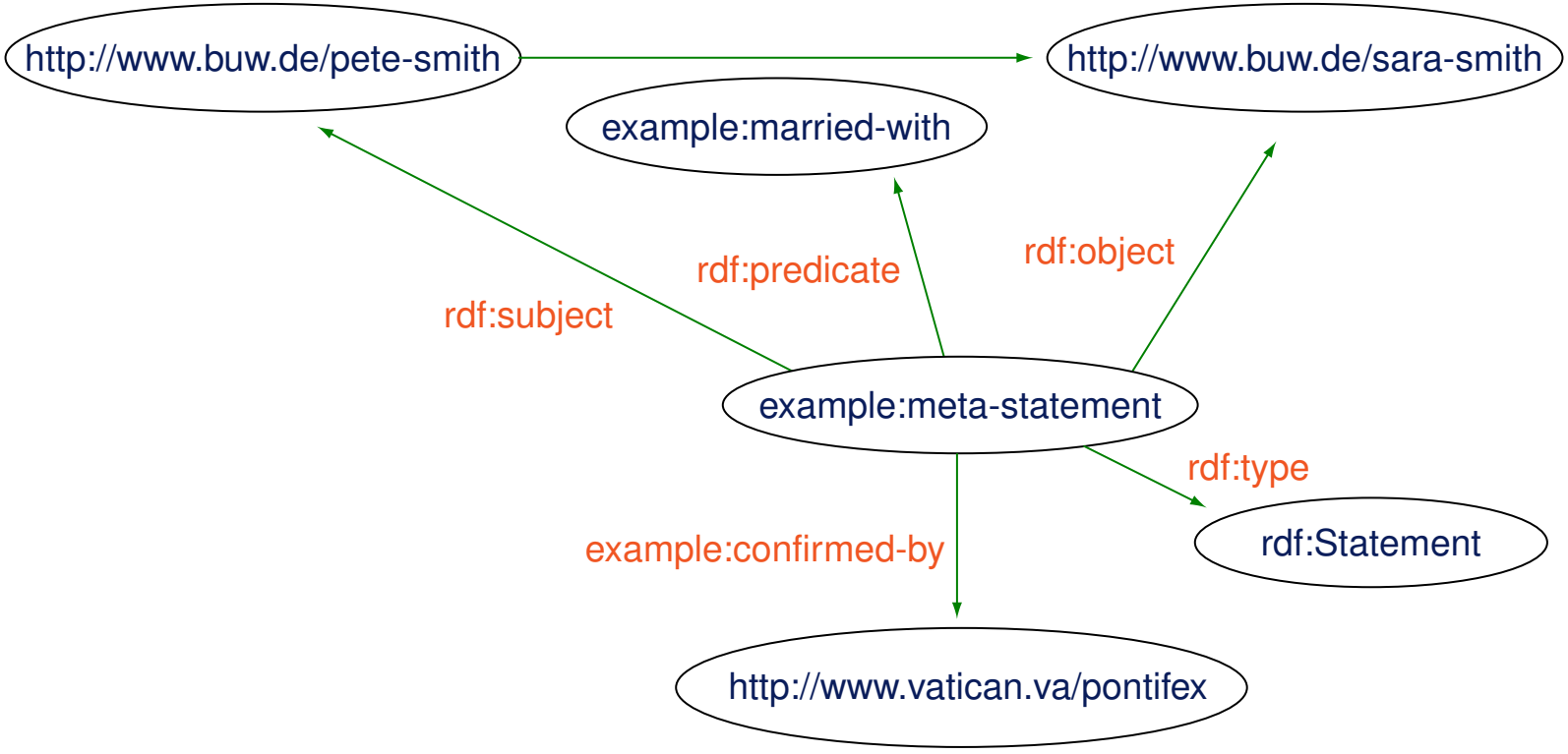
Idee: Für ein Statement wird eine neue Ressource erstellt, die wiederum in einem anderen Statement verwendet werden kann.



RDF: Konzepte

Reification [Serialization]

Idee: Für ein Statement wird eine neue Ressource erstellt, die wiederum in einem anderen Statement verwendet werden kann.



Bemerkungen:

- Reification ist ein Mechanismus, um *Metawissen* zu formulieren.
- Reification könnte man mit „Vergegenständlichung“ übersetzen: Ein Statement wird vergegenständlicht in dem Sinne, dass es Subjekt oder Objekt eines anderen Statements wird.

RDF: XML-Syntax

Definition 1 (RDF Document)

An RDF document is a serialization of an RDF graph into a concrete syntax.

[W3C 2004 www.w3.org/TR/rdf-syntax-grammar]

RDF: XML-Syntax

Definition 1 (RDF Document)

An RDF document is a serialization of an RDF graph into a concrete syntax.

[W3C 2004 www.w3.org/TR/rdf-syntax-grammar]

RDF/XML ist *eine* Möglichkeit zur Serialisierung. W3C-Syntax:

- Ein RDF-Dokument besteht aus einem `<rdf:RDF>`-Element.
- Das `<rdf:RDF>`-Element enthält `<rdf:Description>`-Elemente.
- `<rdf:Description>`-Elemente *(Node Elements)* beschreiben Ressourcen.
- `<rdf:Description>`-Elemente können <*Property*>-Elemente enthalten.
- <*Property*>-Elemente wiederum können ein `<rdf:Description>`-Element oder ein Literal enthalten.

Der Abstieg in `<rdf:Description>`-Elemente entspricht dem Entlanggehen eines Pfades im RDF-Graph.

Bemerkungen:

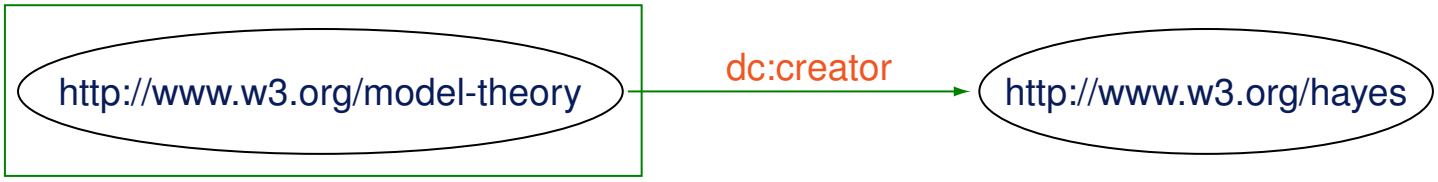
- Verwendete Namensräume (`rdf:`, `dc:`, etc.) müssen deklariert sein.
- Lange Zeichenketten lassen sich mit `Entity`-Deklarationen abkürzen.
- Es gibt eine Standardschreibweise und wenige syntaktische Varianten.
- Die XML-Serialisierung fügt dem RDF-Datenmodell (natürlich) keine weitere Semantik hinzu.

RDF: XML-Syntax

Prinzip

RDF: XML-Syntax

Prinzip

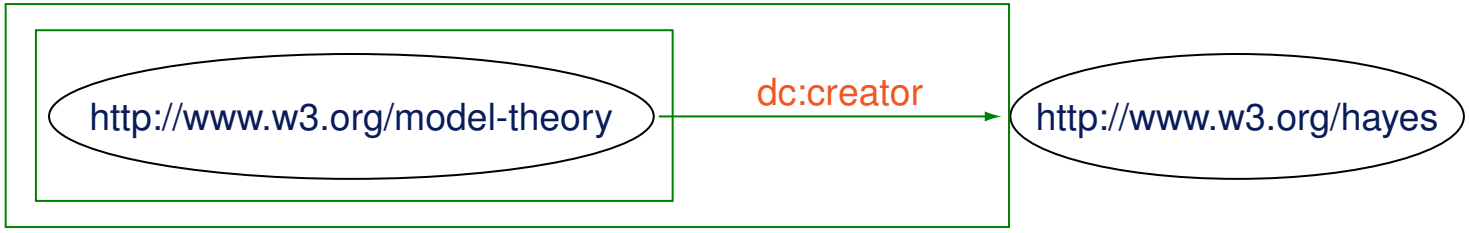



```
<rdf:Description rdf:about="http://www.w3.org/model-theory">

</rdf:Description>
```

RDF: XML-Syntax

Prinzip

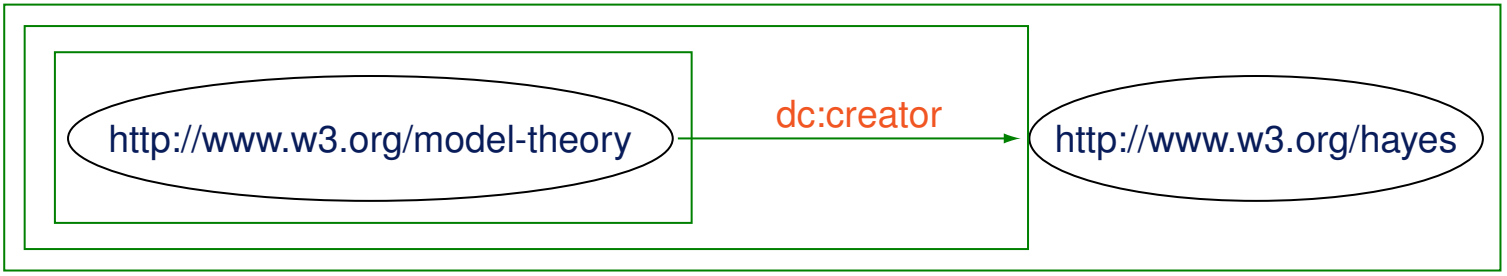



```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
   <dc:creator>


   </dc:creator>
</rdf:Description>
```

RDF: XML-Syntax

Prinzip




```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
   <dc:creator>
      <rdf:Description rdf:about="http://www.w3.org/hayes">
      </rdf:Description>
   </dc:creator>
</rdf:Description>
```

RDF: XML-Syntax

Objekt: Serialisierung als Kindelement oder Attribut

(a) Objekt als `<rdf:Description>`-Kindelement im Property-Element:

```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator>
    <rdf:Description rdf:about="http://www.w3.org/hayes">
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

RDF: XML-Syntax

Objekt: Serialisierung als Kindelement oder Attribut

(a) Objekt als `<rdf:Description>`-Kindelement im Property-Element:

```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator>
    <rdf:Description rdf:about="http://www.w3.org/hayes">
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

≈ (b) Objekt als `rdf:resource`-Attribut im Property-Element:

```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator rdf:resource="http://www.w3.org/hayes" />
</rdf:Description>
```

RDF: XML-Syntax

Objekt: Ressource oder Literal

Objekt ist eine Ressource (eindeutig):


```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator>
    <rdf:Description rdf:about="http://www.w3.org/hayes">
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

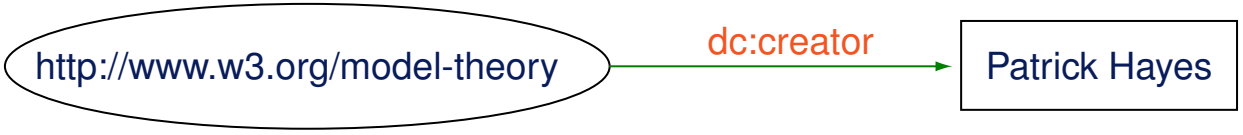
RDF: XML-Syntax

Objekt: Ressource oder Literal

Objekt ist eine Ressource (eindeutig):


```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator>
    <rdf:Description rdf:about="http://www.w3.org/hayes">
    </rdf:Description>
  </dc:creator>
</rdf:Description>
```

Objekt ist ein Literal (muss nicht eindeutig sein):




```
<rdf:Description rdf:about="http://www.w3.org/model-theory">
  <dc:creator>Patrick Hayes</dc:creator>
</rdf:Description>
```

RDF: XML-Syntax

Gemeinsame Vorgängerknoten

Kanten, die im Graph des RDF-Modells von derselben Ressource ausgehen (= Statements mit demselben Subjekt) können als *einzelne* `<rdf:Description>`-Elemente oder *gemeinsam* serialisiert werden:

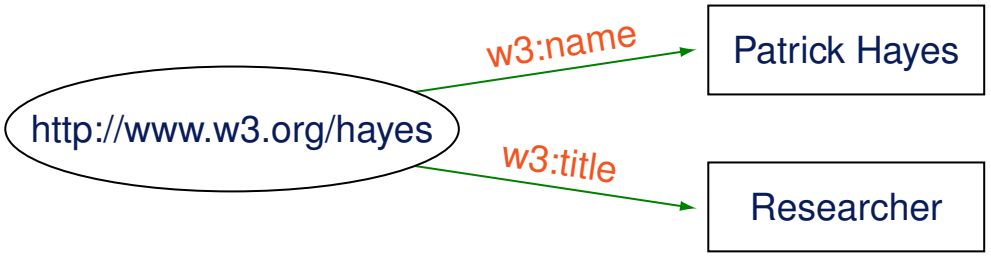
(a)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <w3:name>Patrick Hayes</w3:name>
</rdf:Description>

<rdf:Description rdf:about="http://www.w3.org/hayes">
  <w3:title>Researcher</w3:title>
</rdf:Description>
```

≈ (b)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <w3:name>Patrick Hayes</w3:name>
  <w3:title>Researcher</w3:title>
</rdf:Description>
```

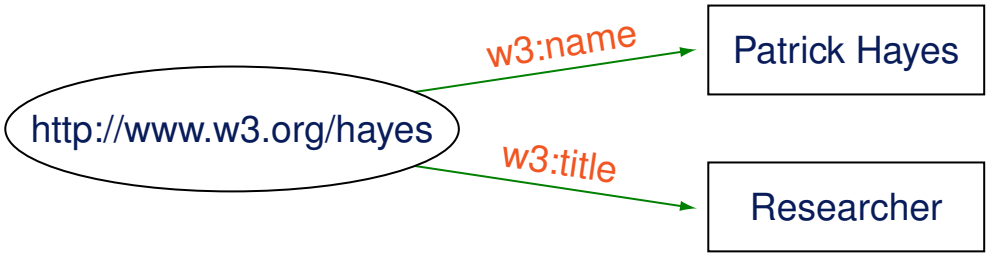
RDF: XML-Syntax

Beispiel



RDF: XML-Syntax

Beispiel




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:w3="http://www.w3.org/">


  <rdf:Description rdf:about="http://www.w3.org/hayes">

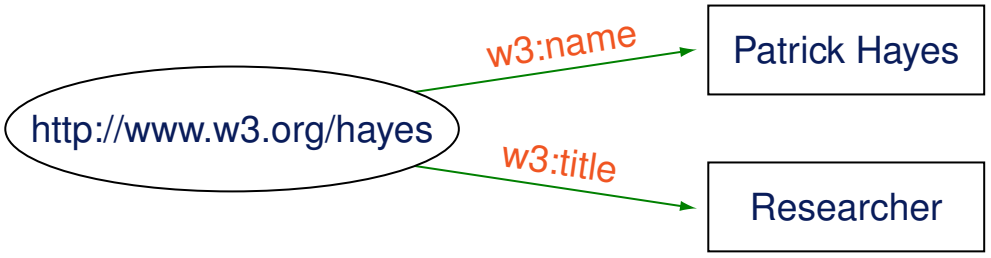
    <w3:name> Patrick Hayes </w3:name>
    <w3:title> Researcher    </w3:title>

  </rdf:Description>


</rdf:RDF>
```

RDF: XML-Syntax

Beispiel




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:w3="http://www.w3.org/">


  <rdf:Description rdf:about="http://www.w3.org/hayes">

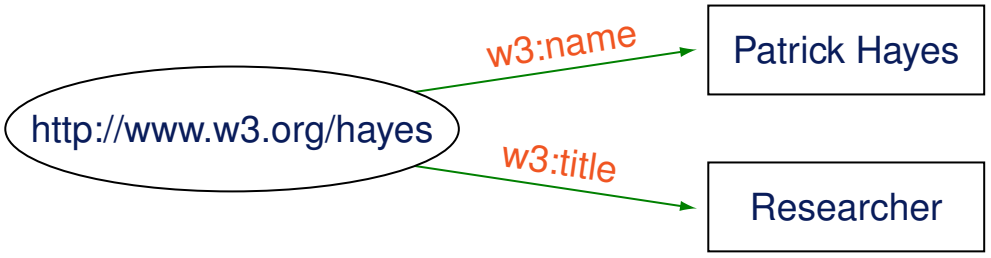
    <w3:name>  Patrick Hayes  </w3:name>
    <w3:title> Researcher     </w3:title>

  </rdf:Description>


</rdf:RDF>
```

RDF: XML-Syntax

Beispiel




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:w3="http://www.w3.org/">

  <rdf:Description rdf:about="http://www.w3.org/hayes">
    <w3:name> Patrick Hayes </w3:name>
    <w3:title> Researcher </w3:title>
  </rdf:Description>

</rdf:RDF>
```

Subjekt

Objekte

Statements

RDF: XML-Syntax

Geltungsbereich

In einem (RDF-)Graph kann man nichts schachteln. Folglich ist der Geltungsbereich *(Scope)* eines `<rdf:Description>`-Elementes immer global.

Die folgenden zwei Deklarationen sind semantisch äquivalent:

(a)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <w3:name>Patrick Hayes</w3:name>
</rdf:Description>

<rdf:Description rdf:about="http://www.buw.de/smith">
  <w3:name>Pete Smith</w3:name>
</rdf:Description>
```

≈ (b)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <w3:name>Patrick Hayes</w3:name>

  <rdf:Description rdf:about="http://www.buw.de/smith">
    <w3:name>Pete Smith</w3:name>
  </rdf:Description>

</rdf:Description>
```

RDF: XML-Syntax

Geltungsbereich (Fortsetzung)

Die folgenden zwei Deklarationen sind semantisch äquivalent:

(a)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <example:knows rdf:resource="http://www.buw.de/smith"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.buw.de/smith">
  <w3:name>Pete Smith</w3:name>
</rdf:Description>
```

RDF: XML-Syntax

Geltungsbereich (Fortsetzung)

Die folgenden zwei Deklarationen sind semantisch äquivalent:

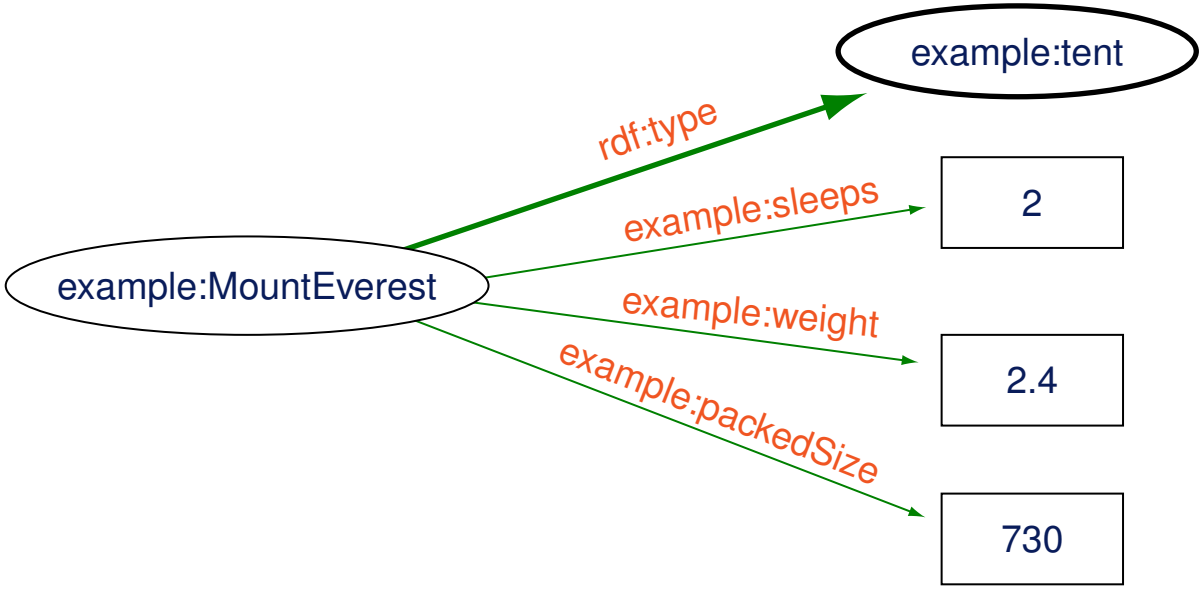
(a)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <example:knows rdf:resource="http://www.buw.de/smith"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.buw.de/smith">
  <w3:name>Pete Smith</w3:name>
</rdf:Description>
```

≈ (b)
```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  <example:knows>
    <rdf:Description rdf:about="http://www.buw.de/smith">
      <w3:name>Pete Smith</w3:name>
    </rdf:Description>
  </example:knows>
</rdf:Description>
```

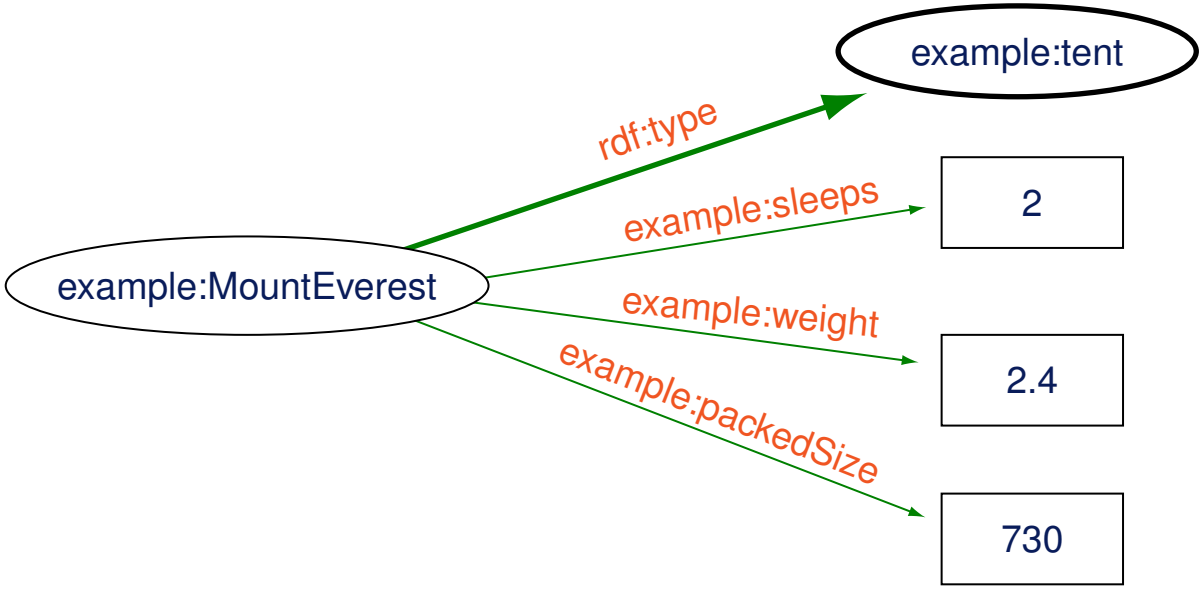
RDF: XML-Syntax

Getypte Ressourcen



RDF: XML-Syntax

Getypte Ressourcen




```
<rdf:Description rdf:about="http://www.buw.de/example/MountEverest">
  <rdf:type rdf:resource="http://www.buw.de/example/Tent"/>
  <example:sleeps>2</example:sleeps>
  <example:weight>2.4</example:weight>
  <example:packedSize>730</example:packedSize>
</rdf:Description>
```

Bemerkungen:

- Der Prefix `example:` steht für die Namensraum-URI `http://www.buw.de/example/`.
- `rdf:type` ist eine Property und ihr Wert kann natürlich auch als Kindelement serialisiert werden:

```
<rdf:Description rdf:about="...">
  </rdf:type>
    <rdf:Description rdf:about="http://www.buw.de/example/Tent"/>
    </rdf:Description>
  </rdf:type>
  <example:sleeps>2</example:sleeps>
  ...
</rdf:Description>
```

RDF: XML-Syntax

Abkürzende Schreibweisen

1. Property-Elemente ohne Kindelemente können im `<rdf:Description>`-Element als XML-Attribut definiert werden.
2. `<rdf:Description>`-Elemente, die eine `<rdf:type>`-Property beinhalten, können durch ein Element mit dem Namen des Typs ersetzt werden. Die `<rdf:type>`-Property kann dann entfallen.

Beispiel:

```
<rdf:Description rdf:about="courseMS21">
  <rdf:type rdf:resource="http://www.buw.de/course"/>
  <example:courseName>WebTec II</example:courseName>
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/l2305"/>
</rdf:Description>
```

RDF: XML-Syntax

Abkürzende Schreibweisen

1. Property-Elemente ohne Kindelemente können im `<rdf:Description>`-Element als XML-Attribut definiert werden.
2. `<rdf:Description>`-Elemente, die eine `<rdf:type>`-Property beinhalten, können durch ein Element mit dem Namen des Typs ersetzt werden. Die `<rdf:type>`-Property kann dann entfallen.

Beispiel:

```
<rdf:Description rdf:about="courseMS21">
  <rdf:type rdf:resource="http://www.buw.de/course"/>
  <example:courseName>WebTec II</example:courseName>
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/l2305"/>
</rdf:Description>
```

≈

```
<rdf:Description rdf:about="courseMS21" example:courseName="WebTec II">
  <rdf:type rdf:resource="http://www.buw.de/course"/>
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/l2305"/>
</rdf:Description>
```

RDF: XML-Syntax

Abkürzende Schreibweisen (Fortsetzung)

1. Property-Elemente ohne Kindelemente können im `<rdf:Description>`-Element als XML-Attribut definiert werden.
2. `<rdf:Description>`-Elemente, die eine `<rdf:type>`-Property beinhalten, können durch ein Element mit dem Namen des Typs ersetzt werden. Die `<rdf:type>`-Property kann dann entfallen.

Beispiel:

```
<rdf:Description rdf:about="courseMS21" example:courseName="WebTec II">
  <rdf:type rdf:resource="http://www.buw.de/course"/>
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/12305"/>
</rdf:Description>
```

RDF: XML-Syntax

Abkürzende Schreibweisen (Fortsetzung)

1. Property-Elemente ohne Kindelemente können im `<rdf:Description>`-Element als XML-Attribut definiert werden.
2. `<rdf:Description>`-Elemente, die eine `<rdf:type>`-Property beinhalten, können durch ein Element mit dem Namen des Typs ersetzt werden. Die `<rdf:type>`-Property kann dann entfallen.

Beispiel:

```
<rdf:Description rdf:about="courseMS21" example:courseName="WebTec II">
  <rdf:type rdf:resource="http://www.buw.de/course"/>
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/12305"/>
</rdf:Description>
```

$\approx$

```
<buw:course rdf:about="courseMS21" example:courseName="WebTec II">
  <example:taughtBy rdf:resource="http://www.buw.de/lecturers/12305"/>
</buw:course>
```

Bemerkungen:

- Der Prefix `buw:` steht für die Namensraum-URI `http://www.buw.de/`.

RDF: XML-Syntax

Abkürzende Schreibweisen bei Containern [RDF-Graph]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF
   [<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <rdf:Description rdf:about="students">
    <rdf:type rdf:resource="&rdf;Bag"/>
    <rdf:li rdf:resource="Adrian"/>
    ...
    <rdf:li rdf:resource="Peter"/>
  </rdf:Description>

</rdf:RDF>
```

RDF: XML-Syntax

Abkürzende Schreibweisen bei Containern [RDF-Graph]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF
   [<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="students">
    <rdf:type rdf:resource="&rdf;Bag"/>
    <rdf:li rdf:resource="Adrian"/>
    ...
    <rdf:li rdf:resource="Peter"/>
  </rdf:Description>
</rdf:RDF>
```

≈

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Bag rdf:about="students">
    <rdf:li rdf:resource="Adrian"/>
    ...
    <rdf:li rdf:resource="Peter"/>
  </rdf:Bag>
</rdf:RDF>
```

RDF: XML-Syntax

Reification [RDF-Graph]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF [<!ENTITY example "http://www.buw.de/example/">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:example="http://www.buw.de/example/">

  <rdf:Description rdf:about="http://www.buw.de/pete-smith">
    <example:married-with rdf:resource="http://www.buw.de/sara-smith"/>
  </rdf:Description>
```

RDF: XML-Syntax

Reification [RDF-Graph]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF [<!ENTITY example "http://www.buw.de/example/">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:example="http://www.buw.de/example/">

  <rdf:Description rdf:about="http://www.buw.de/pete-smith">
    <example:married-with rdf:resource="http://www.buw.de/sara-smith"/>
  </rdf:Description>

  <rdf:Statement rdf:about="&example;meta-statement">
    <rdf:subject rdf:resource="http://www.buw.de/pete-smith"/>
    <rdf:predicate rdf:resource="&example;married-with"/>
    <rdf:object rdf:resource="http://www.buw.de/sara-smith"/>
    <example:confirmed-by rdf:resource="http://www.vatican.va/pontifex"/>
  </rdf:Statement>

</rdf:RDF>
```

RDF: XML-Syntax

Referenzierung versus Definition

- Referenzierung einer existierenden Ressource:

```
<rdf:Description rdf:about="http://www.w3.org/hayes">
  ...
</rdf:Description>
```

- Definition einer neuen Ressource (in einer XML-Datei):

```
<rdf:Description rdf:ID="hayes">
  ...
</rdf:Description>
```

- Referenzierung einer in derselben XML-Datei definierten Ressource:

```
<rdf:Description rdf:about="#hayes">
  ...
</rdf:Description>
```

- Anonyme Ressource *(Blank Node)*:

```
<rdf:Description>
  ...
</rdf:Description>
```

Bemerkungen:

- Die Attribute `rdf:about` und `rdf:ID` besitzen nur deklarativen Charakter innerhalb eines RDF-Modells:

  > *"Formally speaking, the rdf:about-attribute is equivalent to the rdf:ID-attribute: A set of RDF statements together simply form a graph, and there is no such thing as defining an object in one place and referring to it elsewhere."*
  >
  > [p.71, Antoniou/Harmelen 2004]

  Es ist Aufgabe der Anwendung, die ein RDF-Modell verarbeitet, die intendierte Semantik zu operationalisieren. Zum Beispiel lässt der RDF-Validator des W3C `www.w3.org/RDF/Validator` keine zwei gleichen IDs in einem RDF-Dokument zu.

- Enthält das `rdf:about`-Attribut keine URL, sondern nur einen XML-Namen, wird dieser um die Base-URL des Dokuments erweitert. Folglich referenzieren folgende Tags unterschiedliche Ressourcen:

```
<rdf:Description rdf:about="hayes">
                    ≠
<rdf:Description rdf:about="#hayes">
```

RDF: XML-Syntax

Serialisierungssyntax in EBNF

```
RDF ::=              ['<rdf:RDF>'] description* ['</rdf:RDF>']
description ::=      '<rdf:Description' idAboutAttr? '>' propertyElt*
                        '</rdf:Description>'

idAboutAttr ::=      idAttr | aboutAttr
aboutAttr ::=        'about="' URIreference '"'
idAttr ::=           'ID="' IDsymbol '"'
propertyElt ::=      '<' propName '>' value '</' propName '>'
                        | '<' propName resourceAttr '/>'

propName ::=         Qname
value ::=            description | string
resourceAttr ::=     'resource="' URIreference '"'
Qname ::=            [ NSprefix ':' ] name


URIreference ::=     String, interpreted as URI.
IDsymbol, name ::=   Any legal XML name symbol.
NSprefix ::=         Any legal XML namespace prefix.
string ::=           Any XML text, with "<", ">", and "&" escaped.
```

RDF

Quellen zum Nachlernen und Nachschlagen im Web

- W3C RDF-Übersichtseite. www.w3.org/RDF
- D. Becketts RDF-Übersichtseite. planetrdf.com/guide
- F. Manola, E. Miller, Eds. *RDF Primer.*
  W3C Recommendation. www.w3.org/TR/rdf-primer
- ➜ G. Klyne, J. Carroll, Eds. *RDF: Concepts and Abstract Syntax.*
  W3C Recommendation. www.w3.org/TR/rdf-concepts
- D. Beckett, Ed. *RDF/XML Syntax Specification (Revised).*
  W3C Recommendation. www.w3.org/TR/rdf-syntax-grammar
- P. Hayes, Ed. *RDF Semantics.*
  W3C Recommendation. www.w3.org/TR/rdf-mt
- D. Brickley, R.V. Guha, Eds. *RDF Vocabulary Description Language 1.0: RDF Schema.*
  W3C Recommendation. www.w3.org/TR/rdf-schema
- R. Iannella. *An Idiot's Guide to the Resource Description Framework.*
  archive.dstc.edu.au/RDU/reports/RDF-Idiot

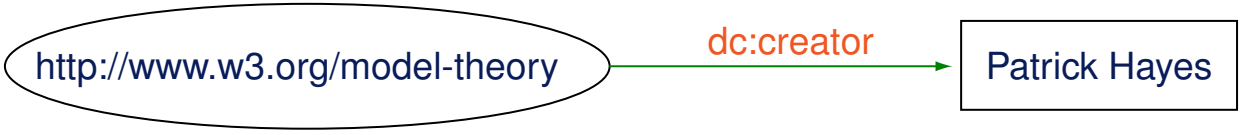
Kapitel WT: VIII (Fortsetzung)

VIII. Semantic Web

- WWW heute
- Semantic Web Vision
- RDF: Einführung
- RDF: Konzepte
- RDF: XML-Serialisierung
- RDF: Anwendungen
- RDFS: Einführung
- RDFS: Konzepte
- Semantik im Web
- Semantik von RDF/RDFS
- Ontologien
- OWL: Konzepte
- OWL: Logikhintergrund
- OWL: Anwendungen

RDFS: Einführung

Typsystem



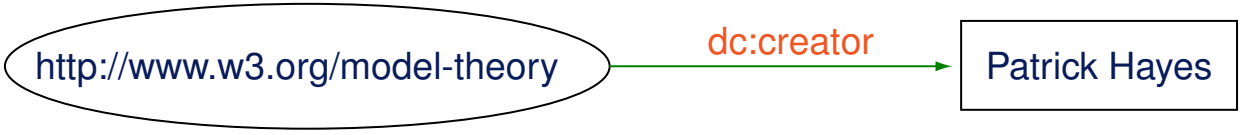
RDF-Properties werden auf zwei Arten benutzt:

1. als Attribute zur Beschreibung einer Ressource
2. zur Charakterisierung von Beziehungen zwischen Ressourcen

Es gibt kein Konzept, um *übergeordnete* Aussagen für Ressourcen zu notieren.

RDFS: Einführung

Typsystem



RDF-Properties werden auf zwei Arten benutzt:

1. als Attribute zur Beschreibung einer Ressource
2. zur Charakterisierung von Beziehungen zwischen Ressourcen

Es gibt kein Konzept, um *übergeordnete* Aussagen für Ressourcen zu notieren.

➜ RDF-Schema (RDFS, RDF vocabulary description language):

- dient zur Erstellung von Vokabularbeschreibungen
- ermöglicht die Gruppierung von zusammengehörenden Ressourcen
- ermöglicht die Beschreibung von Eigenschaften zwischen Ressourcen

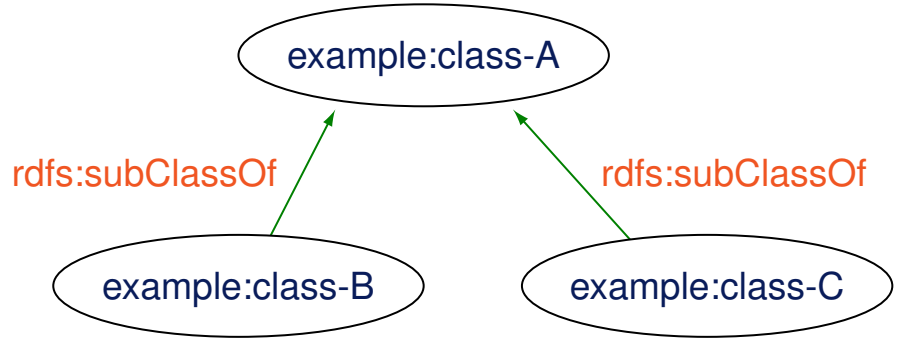
RDF-Schema stellt ein Typsystem für RDF zur Verfügung.

RDFS: Einführung

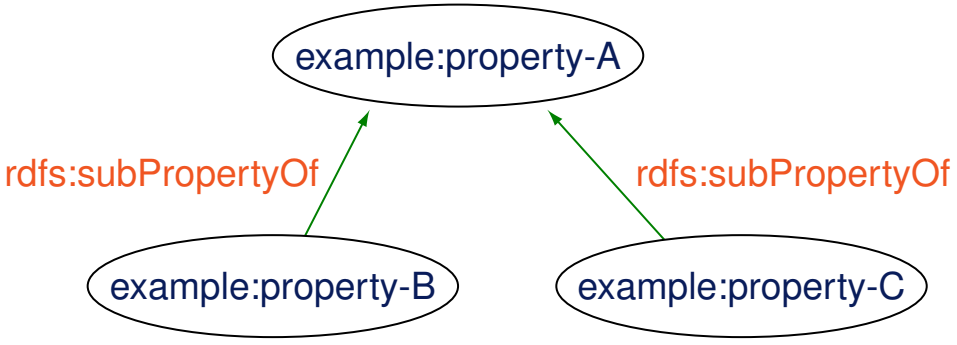
Typsystem

Zusammenfassung von Ressourcen mit gleichen Eigenschaften. Unterklassen und Unter-Properties sind Teilmengen ihrer Oberklasse bzw. Ober-Properties.

1. Klassenhierarchie:



2. Property-Hierarchie:



Bemerkungen:

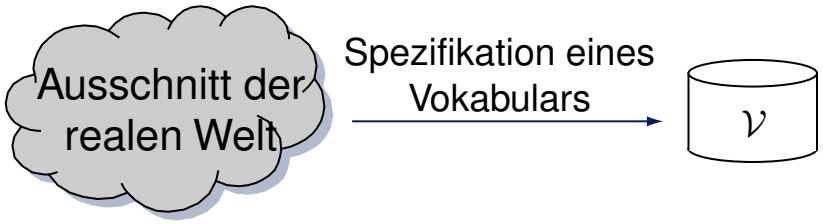
- Vokabularbeschreibungen für RDF-Schema sind in RDF, also unter Rückgriff auf das Datenmodell und die Syntax von RDF formuliert.
- Die Vokabulare (einschließlich Kommentar, Angabe der Signatur bei Prädikat-Ressourcen, Angabe der Oberklasse bei Subjekt-Ressourcen) für RDF und RDFS finden sich in den zugehörigen Namensräumen `http://www.w3.org/1999/02/22-rdf-syntax-ns#` bzw. `http://www.w3.org/2000/01/rdf-schema`
- Die Namensräume für RDF und RDFS spezifizieren eine *intendierte* Semantik. Mit `rdf:subClassOf` (zum Beispiel) wird etwas Bestimmtes intendiert bzw. gefordert. Es ist Aufgabe der Anwendung, die ein RDF-Schema verarbeitet, die intendierte Semantik zu operationalisieren.

  Wie die intendierte Semantik *exakt* zu implementieren ist, also welche Eigenschaften und welches Verhalten ein Element des Vokabulars aufzuweisen hat, ist in `http://www.w3.org/TR/rdf-mt/` definiert.

  Zum Beispiel wird durch `rdf:subClassOf` unter anderem gefordert, dass diejenige Klasse, die eine Unterklasse einer Klasse C ist, von dem gleichem Typ wie C ist.
- Letztendlich sollen die in den Vokabularbeschreibungen modellierten Zusammenhänge einem *Schlussfolgerungsprozess* zugänglich gemacht werden. Die Durchführung von Schlussfolgerungsprozessen geschieht in Anwendungsprogrammen.

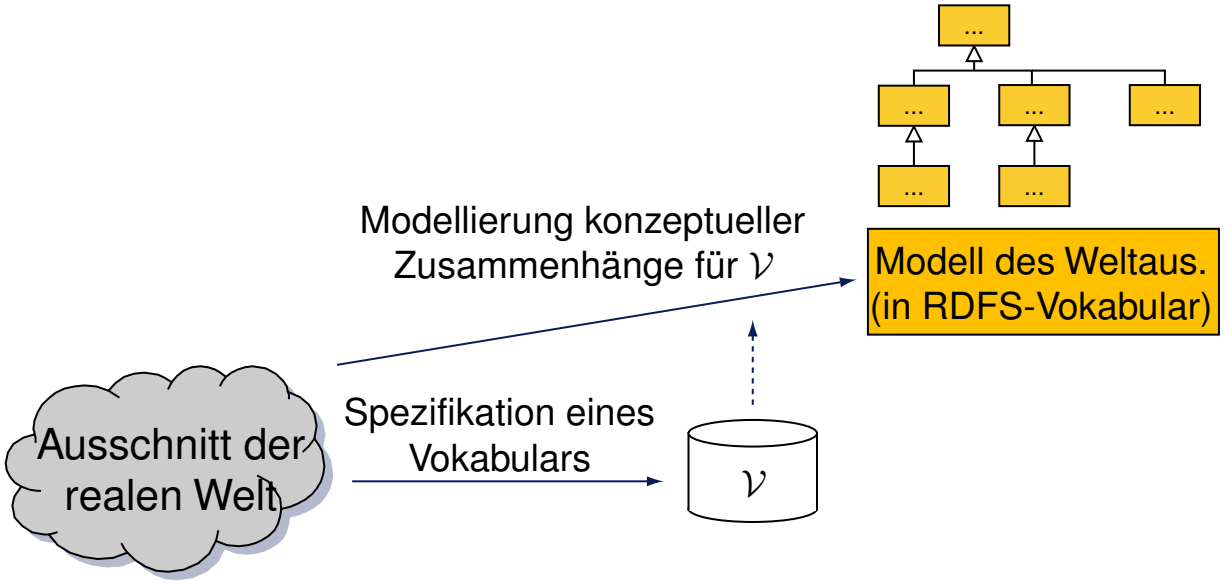
RDFS: Einführung

Modellieren und Schlussfolgern mit RDF/RDFS



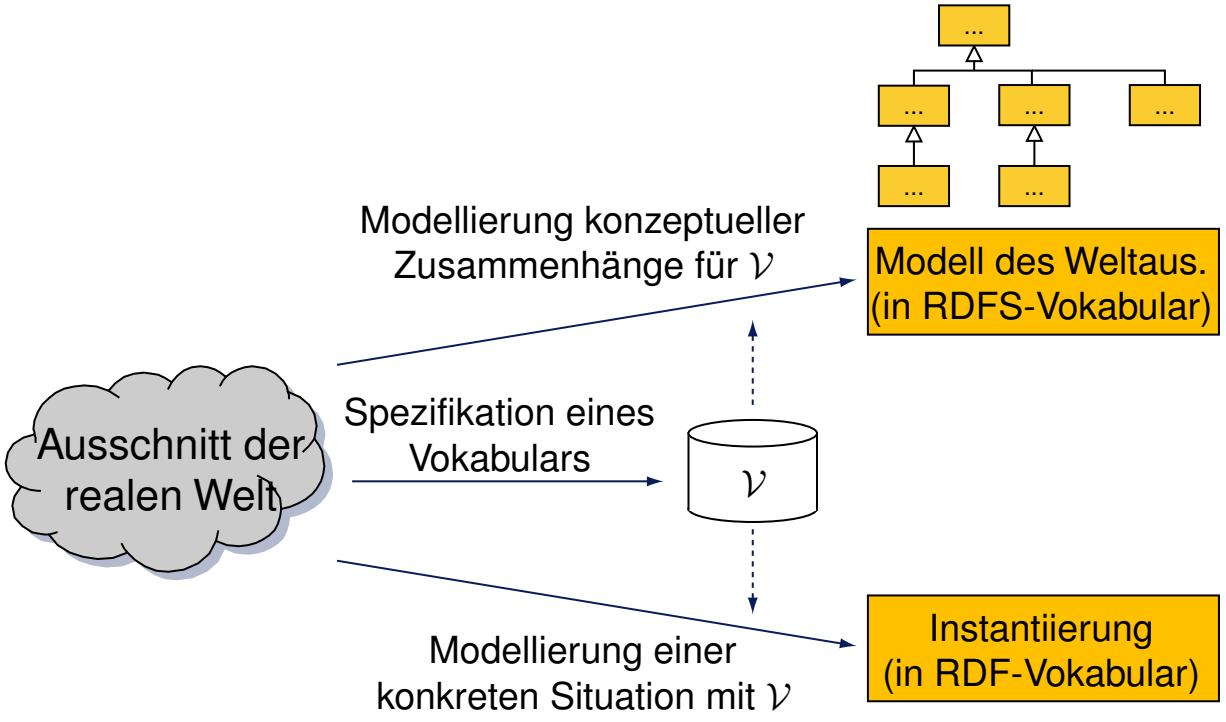
RDFS: Einführung

Modellieren und Schlussfolgern mit RDF/RDFS



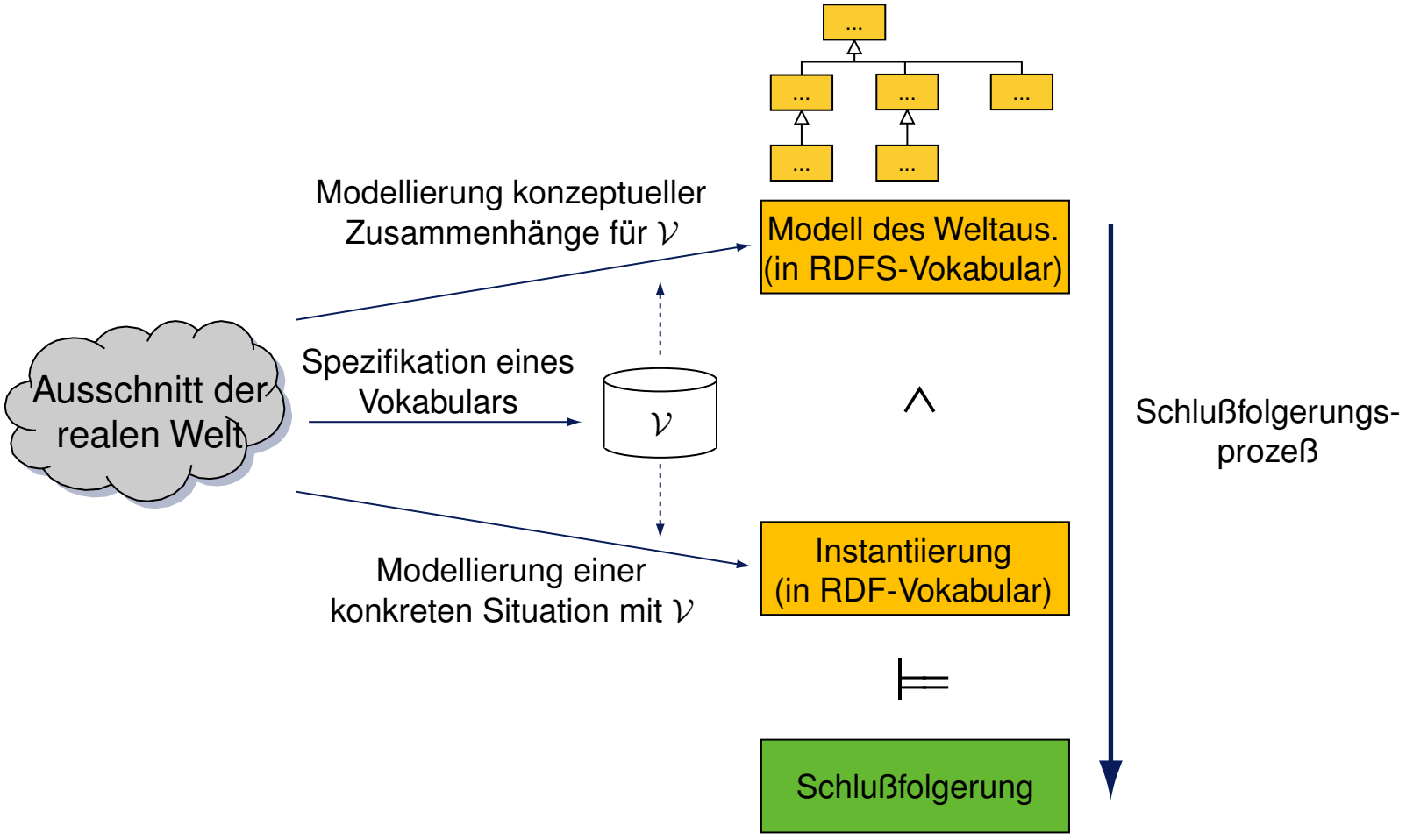
RDFS: Einführung

Modellieren und Schlussfolgern mit RDF/RDFS



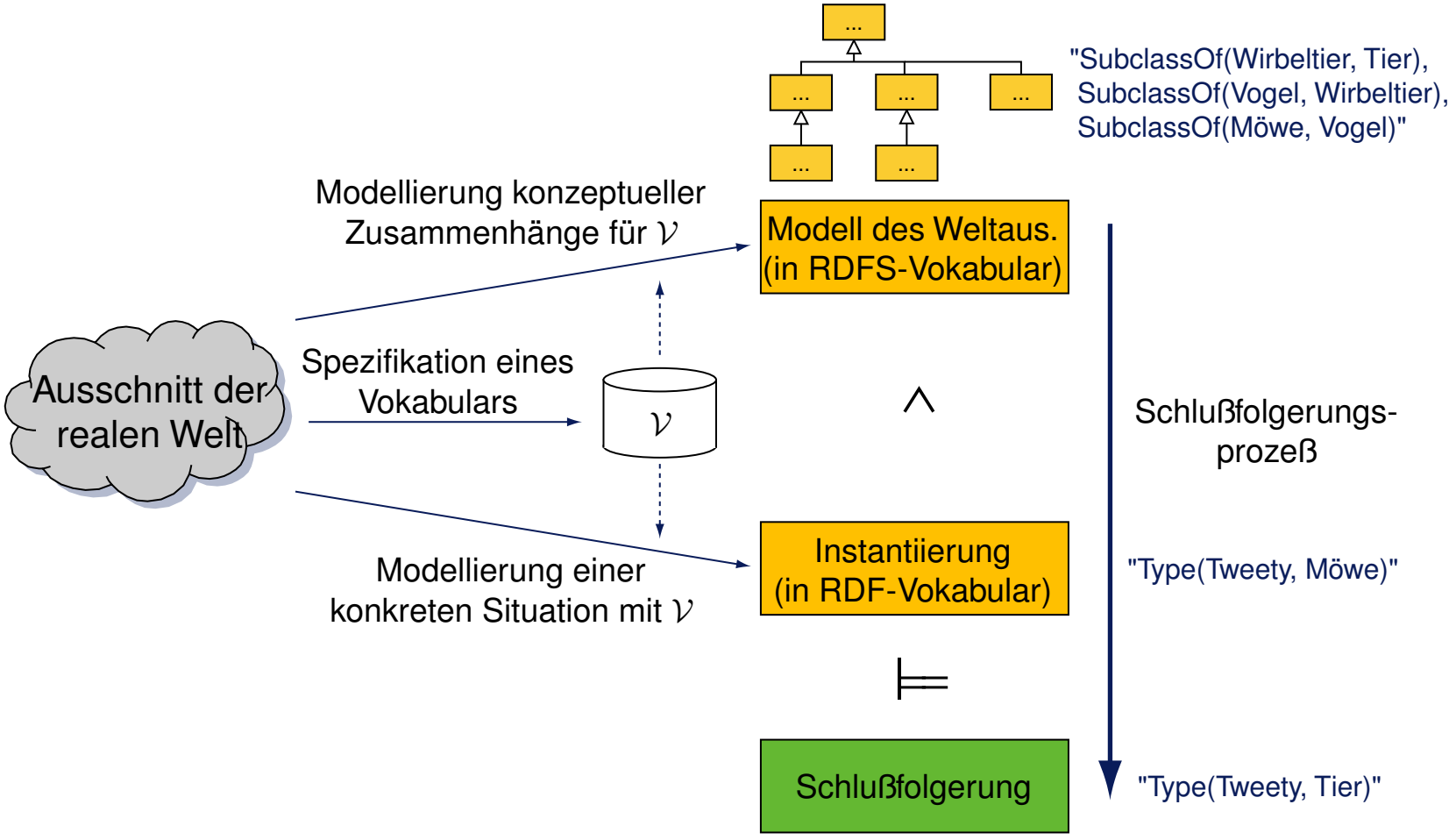
RDFS: Einführung

Modellieren und Schlussfolgern mit RDF/RDFS



RDFS: Einführung

Modellieren und Schlussfolgern mit RDF/RDFS



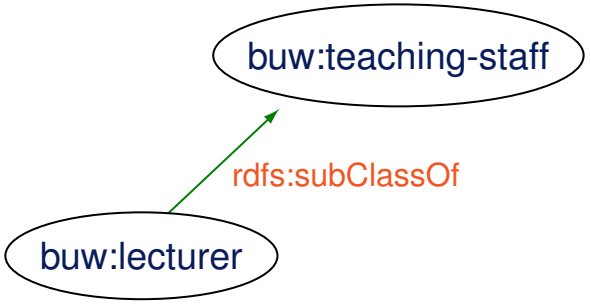
Bemerkungen:

- RDF und RDFS sind formale Sprachen. Sie lassen sich in die Prädikatenlogik einbetten und stellen – in diesem Sinne – Spezialisierungen der Prädikatenlogik dar.
- Das RDFS-Modell des Weltausschnitts kann deshalb unmittelbar als eine Axiomatisierung (= Formel in der Prädikatenlogik + intensionale Interpretation) des Weltausschnitts aufgefasst werden. Eine Instanziierung entspricht dann einer Menge von Grundprädikaten (= Prädikate ohne Variablen).

  Aus dieser Menge von Formeln lassen sich mit Hilfe eines Kalküls Schlussfolgerungen ziehen.
- In der Praxis des Semantic Web ist die Verwendung der vollständigen Sprache der Prädikatenlogik nicht sinnvoll. Deshalb werden bestimmte Einschränkungen gemacht und es kommen spezielle Logiken zum Einsatz. Stichworte: Beschreibungslogik *(Description Logics, DL)*, Frame-Logic

RDFS: Konzepte

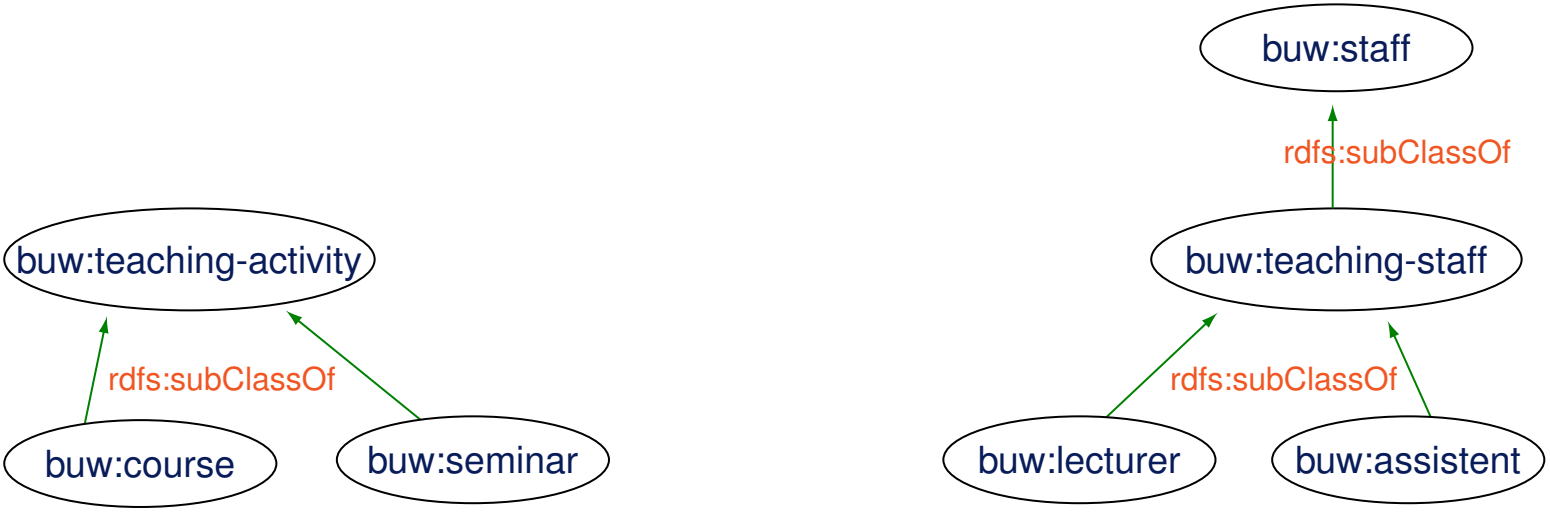
Modellierungsbeispiel




```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

RDFS: Konzepte

Modellierungsbeispiel

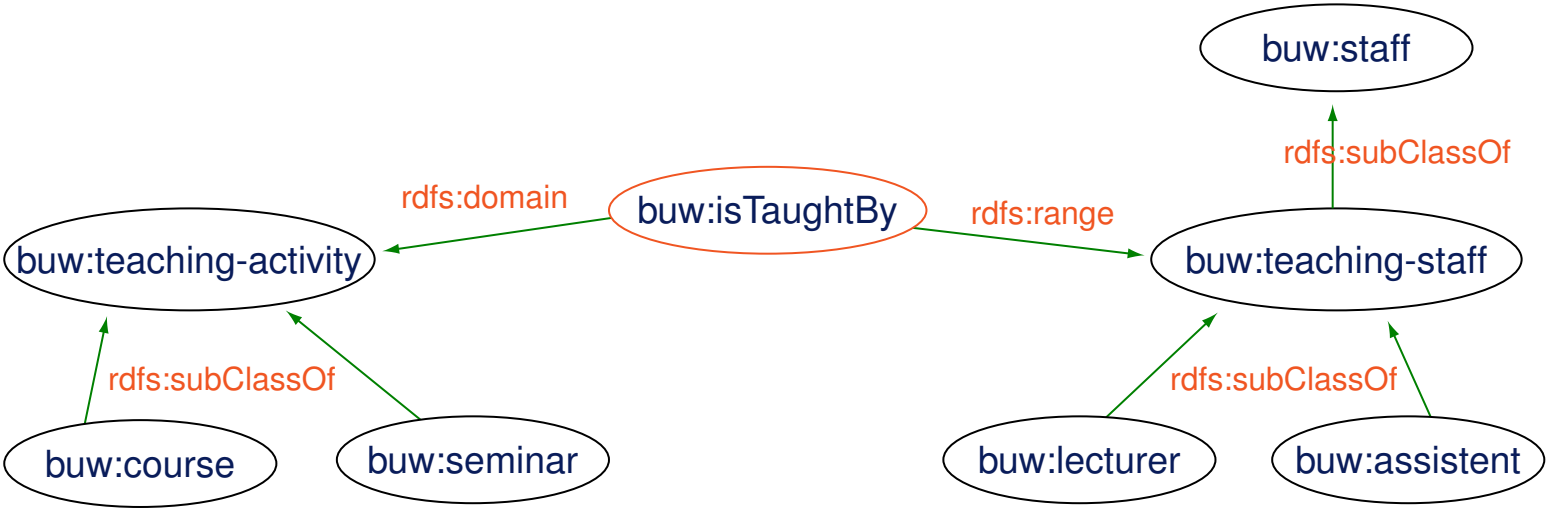



```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>

...
```

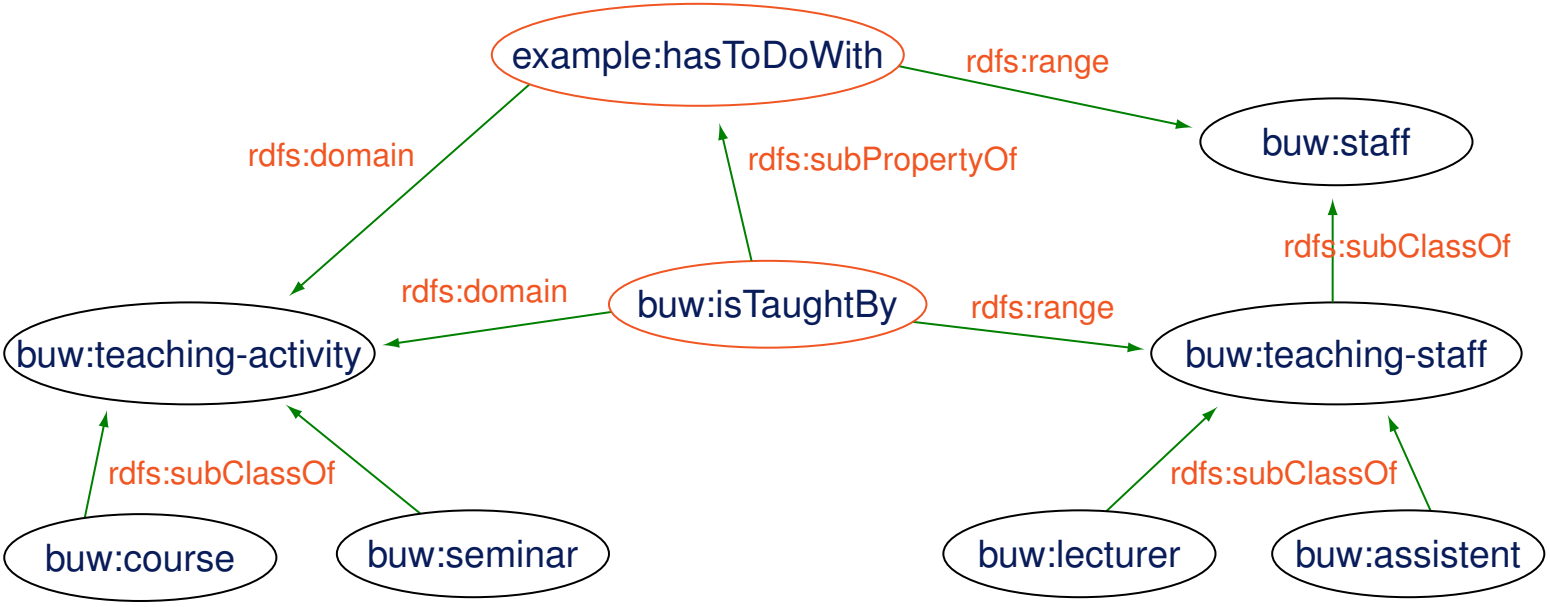
RDFS: Konzepte

Modellierungsbeispiel



RDFS: Konzepte

Modellierungsbeispiel

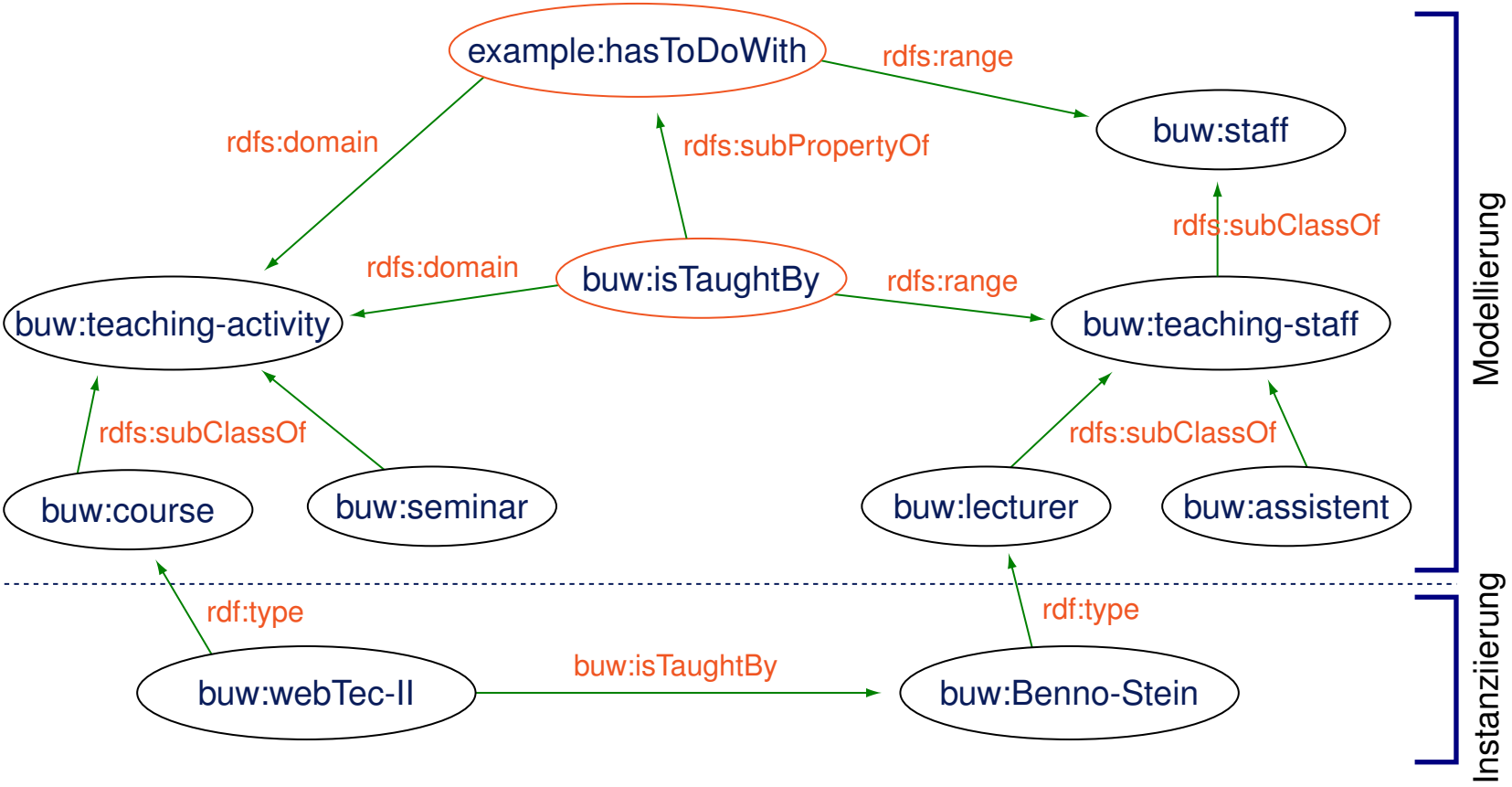



```
<rdfs:Property rdf:about="&example;hasToDoWith">
  <rdfs:domain rdf:resource="http://www.buw.de/teaching-activity"/>
  <rdfs:range rdf:resource="http://www.buw.de/staff"/>
</rdfs:Property>

<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
  <rdfs:range rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Property>
```

RDFS: Konzepte

Modellierungsbeispiel




```
<rdf:Description rdf:about="http://www.buw.de/Benno-Stein">
  < rdf:type rdf:resource="http://www.buw.de/lecturer"/>
</rdf:Description>
```

RDFS: Konzepte

Vokabular: Klassen

Klassenname	Beschreibung
`rdfs:Resource`	die Klasse aller Ressourcen
`rdfs:Class`	die Klasse aller Klassen
`rdf:Property`	die Klasse aller Ressourcen, die Properties sind
`rdfs:Literal`	die Klasse aller String-Literale
`rdf:Statement`	die Klasse aller vergegenständlichten Statements
`rdfs:Container`	die Klasse aller Container-Klassen
`rdf:Bag`	die Klasse der ungeordneten Mengen
`rdf:Seq`	die Klasse der geordneten Mengen
`rdf:Alt`	die Klasse der Alternativen

RDFS: Konzepte

Vokabular: Properties

Property-Name	Domain	Range	Beschreibung
`rdf:type`	`rdfs:Resource`	`rdfs:Class`	Instanzbeziehung
`rdfs:subClassOf`	`rdfs:Class`	`rdfs:Class`	Spezialisierungsbeziehung zwischen Klassen
`rdfs:subPropertyOf`	`rdfs:Property`	`rdfs:Property`	Spezialisierungsbeziehung zwischen Properties
`rdfs:domain`	`rdfs:Property`	`rdfs:Class`	Einschränkung des Urbildbereichs
`rdfs:range`	`rdfs:Property`	`rdfs:Class`	Einschränkung des Bildbereichs
`rdfs:member`	`rdfs:Container`	`rdfs:Class`	Elementbeziehung zu einer Container-Klasse

RDFS: Konzepte

Vokabular: Properties (Fortsetzung)

Property-Name	Domain	Range	Beschreibung
`rdf:subject`	`rdf:Statement`	`rdfs:Resource`	kennzeichnet Ressource als Subjekt eines Statements
`rdf:predicate`	`rdf:Statement`	`rdfs:Property`	kennzeichnet Property als Prädikat eines Statements
`rdf:object`	`rdf:Statement`	`rdfs:Resource`	kennzeichnet Ressource als Objekt eines Statements
`rdfs:seeAlso`	`rdfs:Resource`	`rdfs:Resource`	verweist auf Ressource mit Zusatzinformation
`rdfs:isDefinedBy`	`rdfs:Resource`	`rdfs:Resource`	verweist auf Definitions-URI
`rdfs:comment`	`rdfs:Resource`	`rdfs:Literal`	Kommentar zur Ressource
`rdfs:label`	`rdfs:Resource`	`rdfs:Literal`	verständlicher Ressource-Name

Bemerkungen:

- Die Tabellen zeigen eine Teilmenge des Vokabulars.
- Der Prefix `rdf:` steht für die Namensraum-URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#`; der Prefix `rdfs:` steht für die Namensraum-URI `http://www.w3.org/2000/01/rdf-schema#`. Dort befinden sich die vollständigen Vokabularbeschreibungen der RDF- und RDFS-Ressourcen.
- Eine Klasse kann Unterklasse mehrerer Klassen sein; eine Property kann Unter-Property mehrerer Properties sein.
- Die Semantik der Unterklassen- und Unter-Property-Relationen beinhaltet die Transitivität dieser Relationen.

RDFS: Konzepte

Property-zentrierte Modellierung

Die Begriffe „Klasse“, „Eigenschaft“ oder „Vererbung“ sind Merkmale vieler objektorientierter Sprachen. Abweichend zur verbreiteten Semantik gilt in RDF:

- RDF-Properties sind global sichtbar.
- Eine RDF-Klasse *definiert nicht* und *kapselt nicht* die ihr zugeordneten Properties – sondern: Properties werden Ressourcen zugeordnet.

RDFS: Konzepte

Property-zentrierte Modellierung

Die Begriffe „Klasse“, „Eigenschaft“ oder „Vererbung“ sind Merkmale vieler objektorientierter Sprachen. Abweichend zur verbreiteten Semantik gilt in RDF:

- RDF-Properties sind global sichtbar.
- Eine RDF-Klasse *definiert nicht* und *kapselt nicht* die ihr zugeordneten Properties – sondern: Properties werden Ressourcen zugeordnet.

Klassisch objektorientiert

RDF/RDFS

```
<rdfs:Property rdf:about="&example;author">
  <rdfs:domain rdf:resource="&example;book"/>
  <rdfs:range rdf:resource="&example;person"/>
</rdfs:Property>

<rdf:Description rdf:ID="Pearl-Thesis">
  <rdf:type rdf:resource="&example;book"/>
  <example:Title>Heuristics</example:title>
  <example:Author>Judea Pearl</example:author>
</rdf:Description>
```

RDFS: Konzepte

Property-zentrierte Modellierung

author: book $\longrightarrow$ person

function: domain $\longrightarrow$ range

Modellierung

RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person

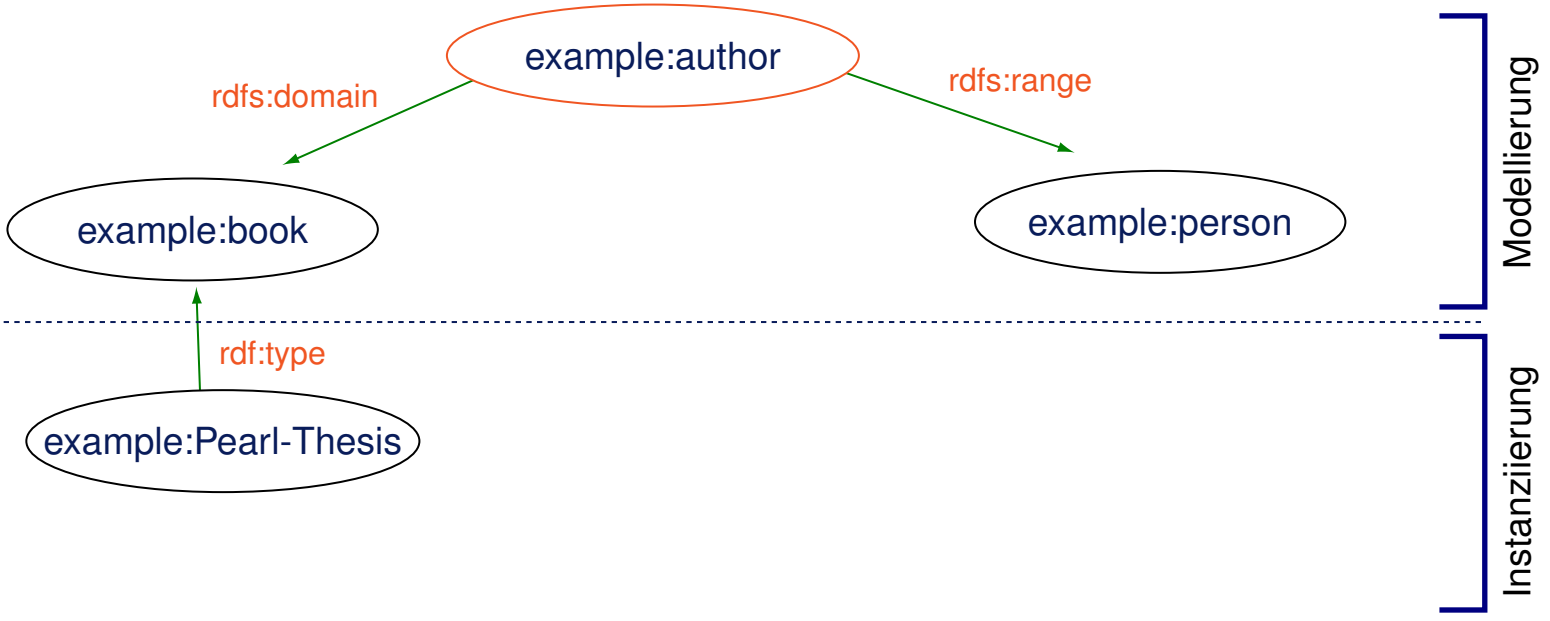
function: domain ⟶ range

RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person

function: domain ⟶ range

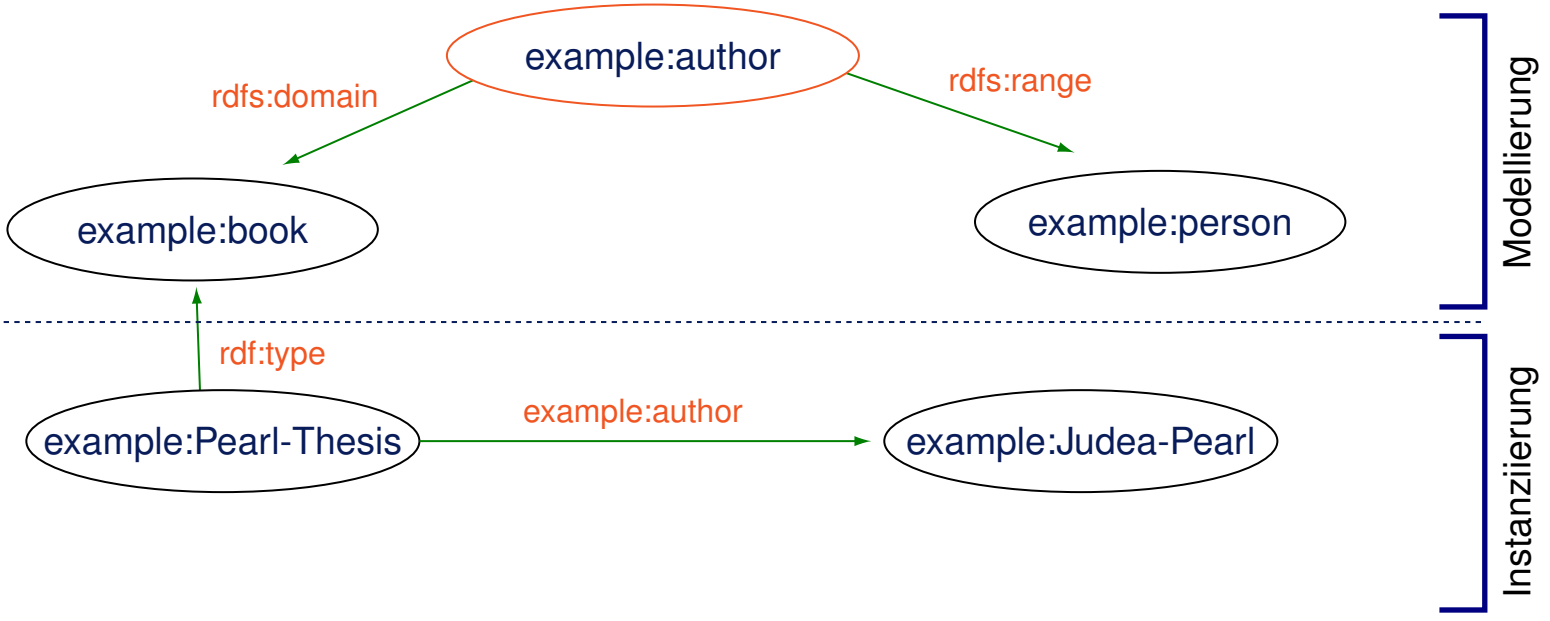


RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person

function: domain ⟶ range

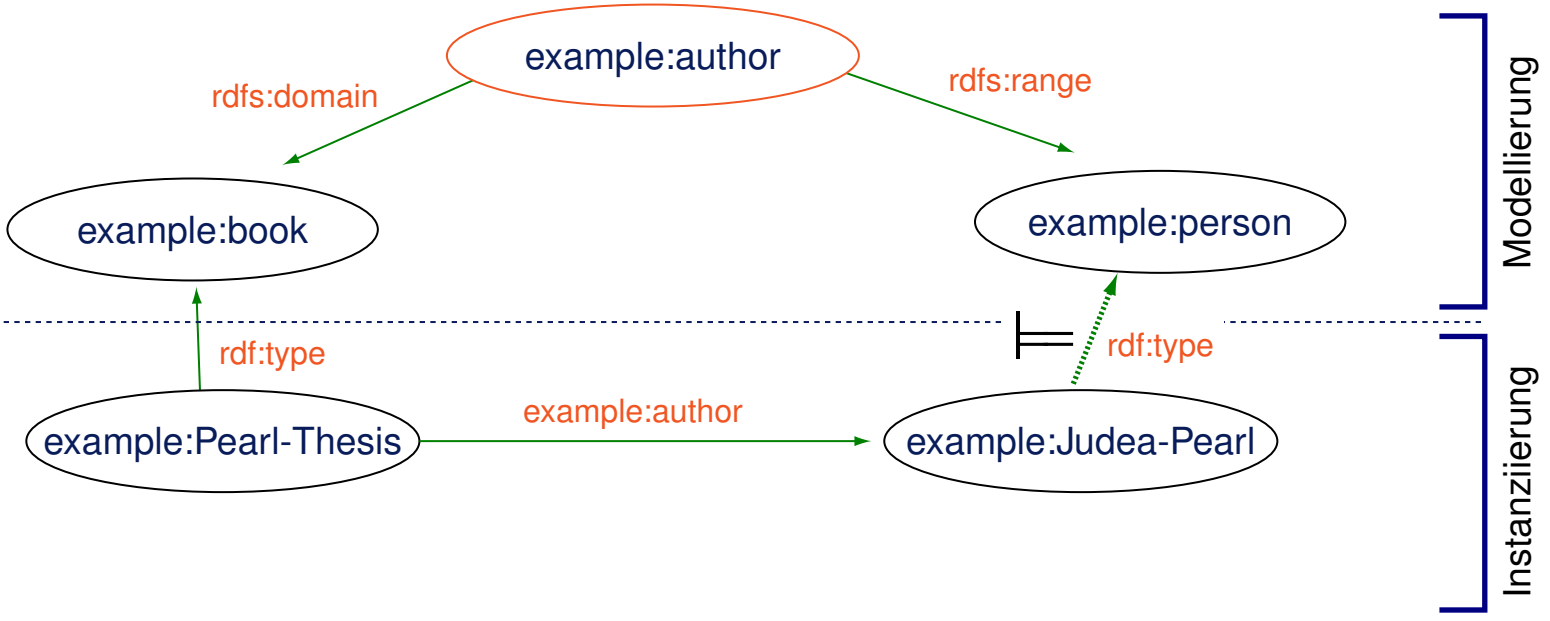


RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person

function: domain ⟶ range

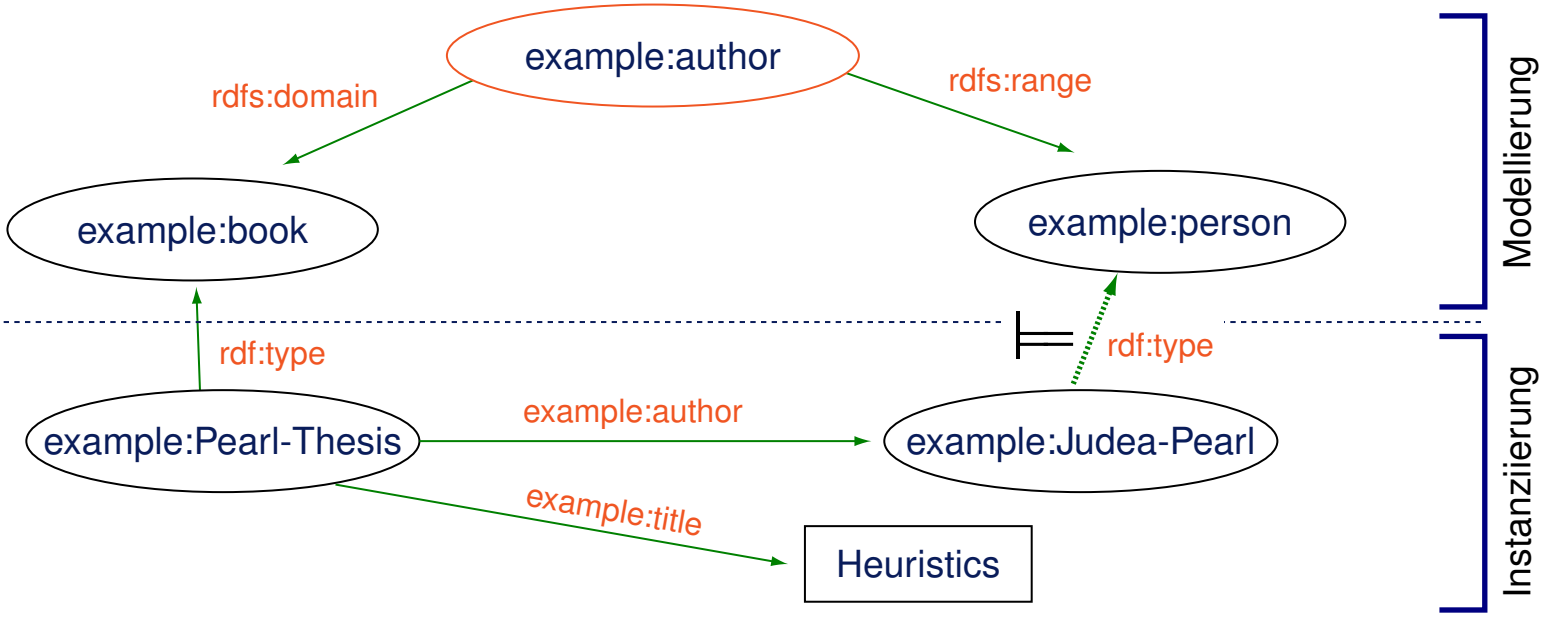


RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person

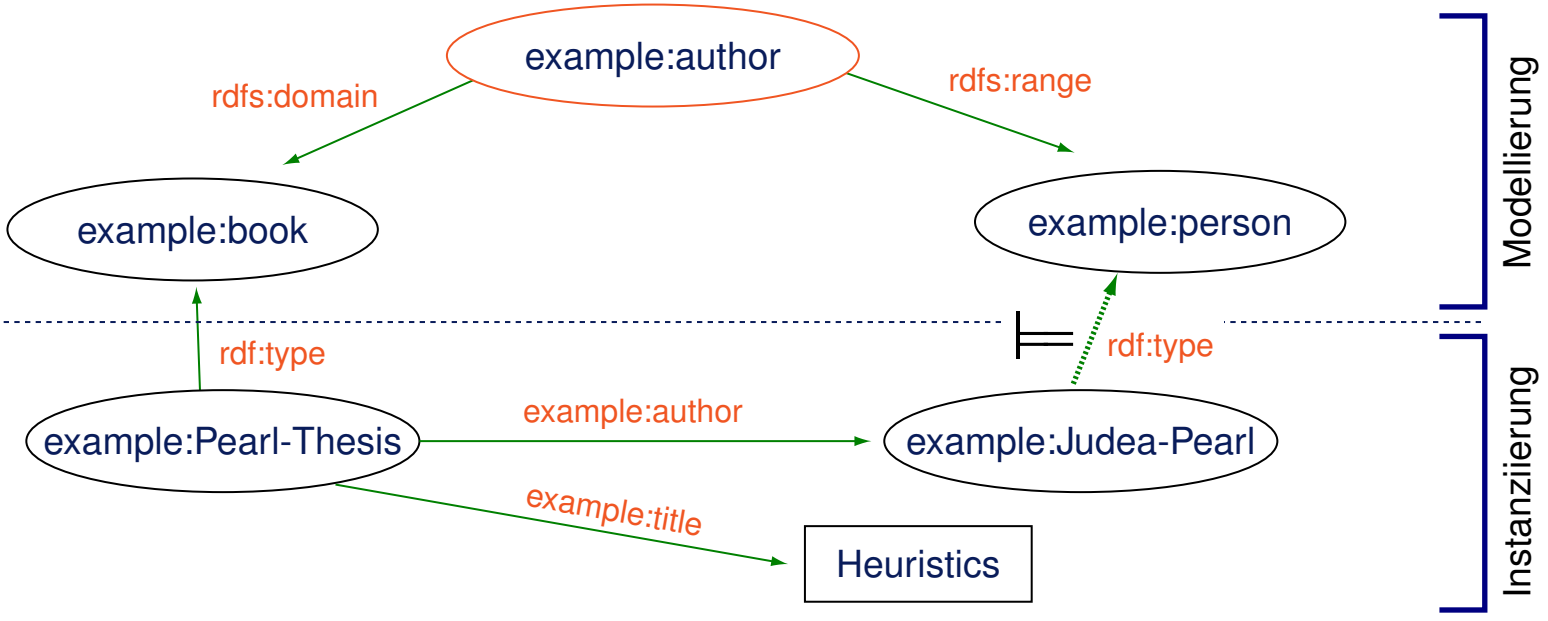
function: domain ⟶ range



RDFS: Konzepte

Property-zentrierte Modellierung

author: book ⟶ person
function: domain ⟶ range



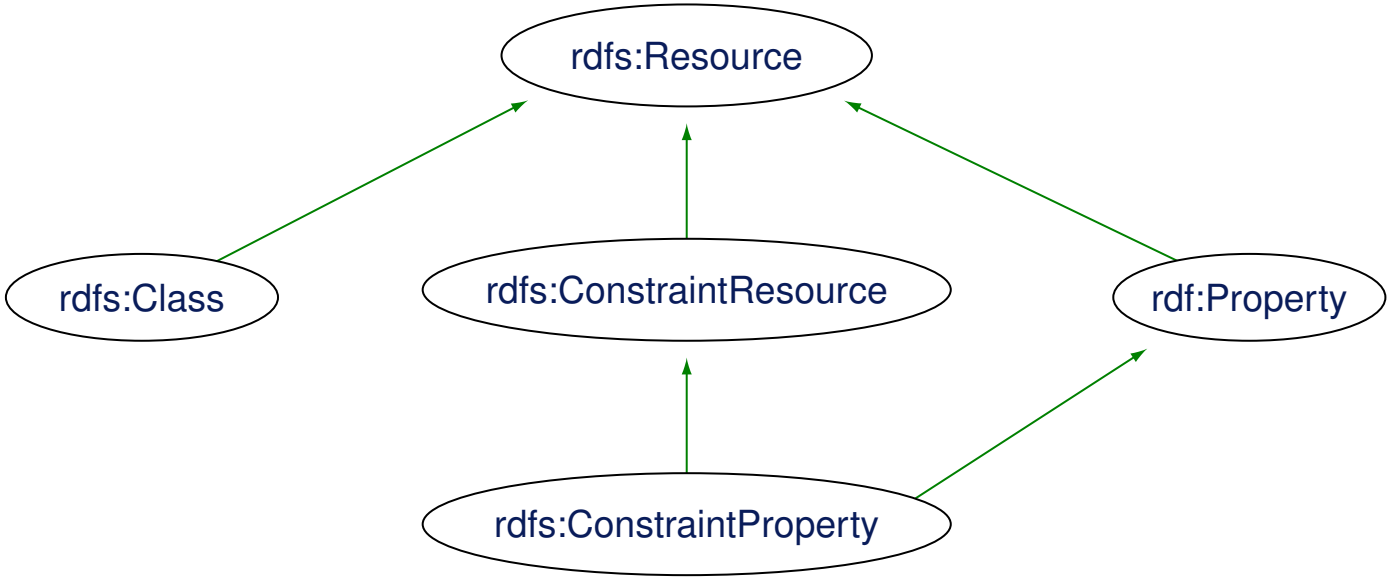
➜ Ressourcen (Klassen) können jederzeit und von jedem erweitert werden.

RDFS: Konzepte

RDFS in RDF

RDF-Schema selbst ist mittels des RDF-Schema-Vokabulars und in dem Datenmodell von RDF definiert.

Ausschnitt der *Klassen*hierarchie:



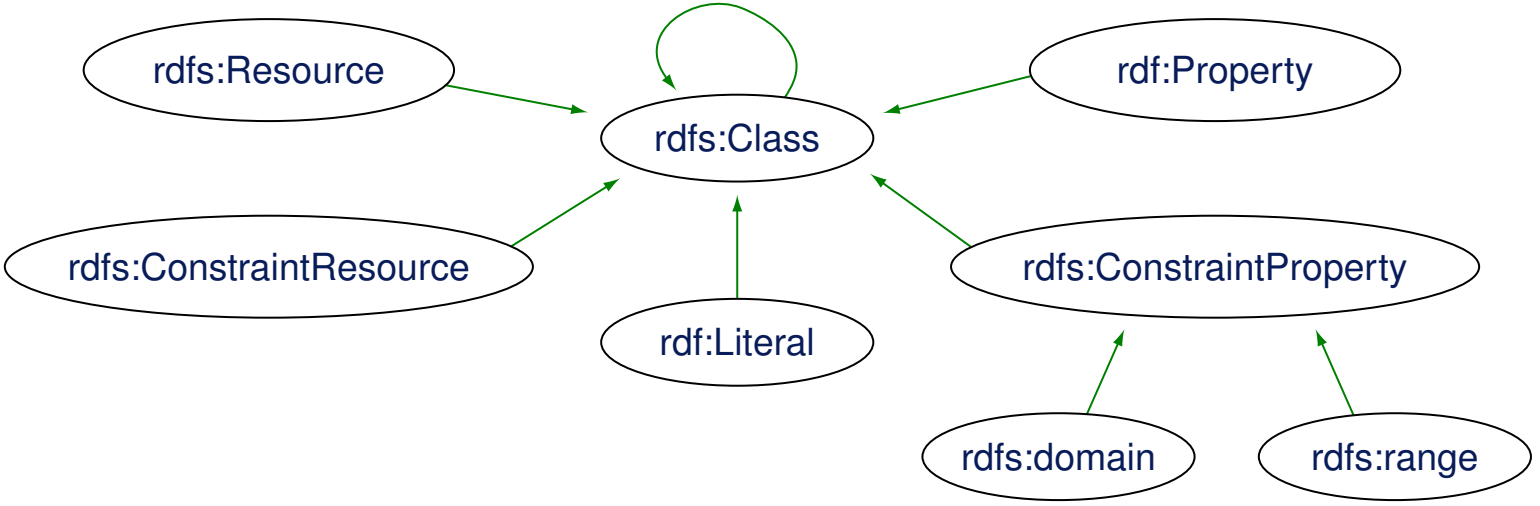
Alle Beziehungen sind vom Typ `rdfs:subClassOf`.

RDFS: Konzepte

RDFS in RDF

RDF-Schema selbst ist mittels des RDF-Schema-Vokabulars und in dem Datenmodell von RDF definiert.

Ausschnitt der *Typ*hierarchie:



Alle Beziehungen sind vom Typ `rdf:type`.

Bemerkungen und Kritikpunkte an RDFS [vgl. Tomczyk 2004]:

- Nur für manche Primitive gibt es eine explizite Semantik.
- Teilweise schwer verständliches formales Modell: eine Ressource kann gleichzeitig eine Instanz (Individuum), eine Klasse (Konzept) und ein Prädikat (Rolle) sein.
- Die Aussdrucksstärke von RDFS ist gering:
  - nur domain/range Einschränkungen von Properties
  - keine einfachen Axiome wie Reflexivität, Symmetrie und Transitivität
  - keine allgemeinen Regeln

Kapitel WT: VIII (Fortsetzung)

VIII. Semantic Web

- WWW heute
- Semantic Web Vision
- RDF: Einführung
- RDF: Konzepte
- RDF: XML-Serialisierung
- RDF: Anwendungen
- RDFS: Einführung
- RDFS: Konzepte
- Semantik im Web
- Semantik von RDF/RDFS
- Ontologien
- OWL: Konzepte
- OWL: Logikhintergrund
- OWL: Anwendungen

Semantik im Web

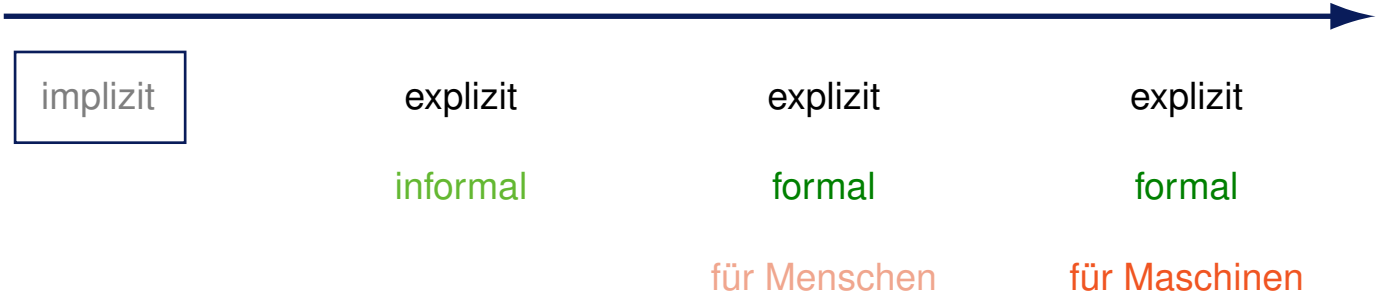
„The Semantic Continuum“

implizit	explizit	explizit	explizit
	informal	formal	formal
		für Menschen	für Maschinen

[vgl. Uschold 2002]

Semantik im Web

„The Semantic Continuum“

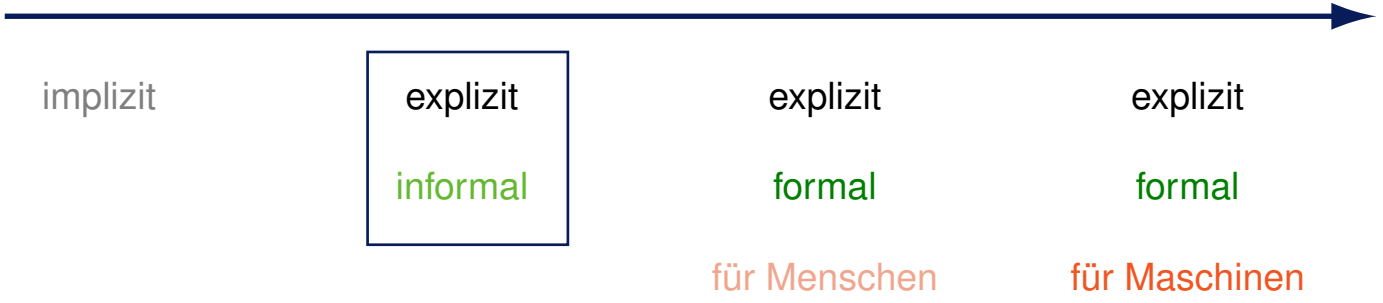


Implizite, kontextabhängige Semantik:

- Beispiel: „sprechende“ XML-Tags

Semantik im Web

„The Semantic Continuum“

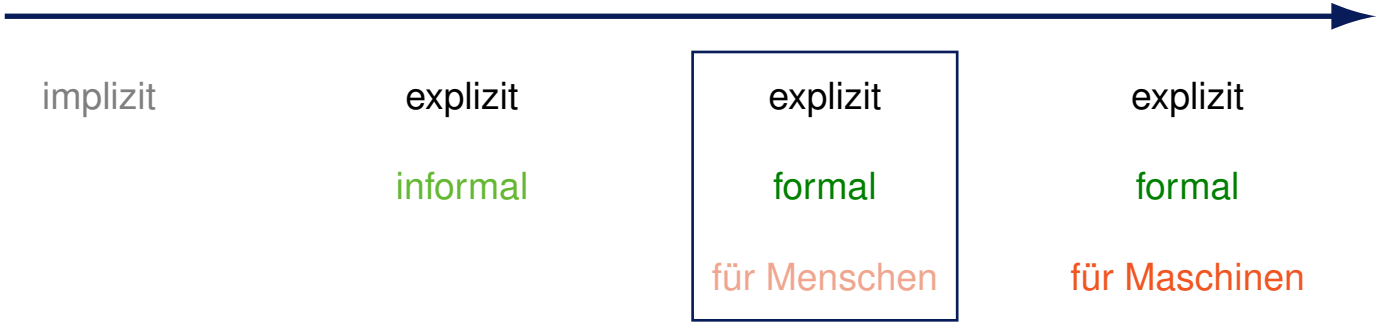


Informale Sprache mit expliziter Semantik:

- Beispiele: Bedienungsanleitung, Glossar, Java-Doc

Semantik im Web

„The Semantic Continuum“

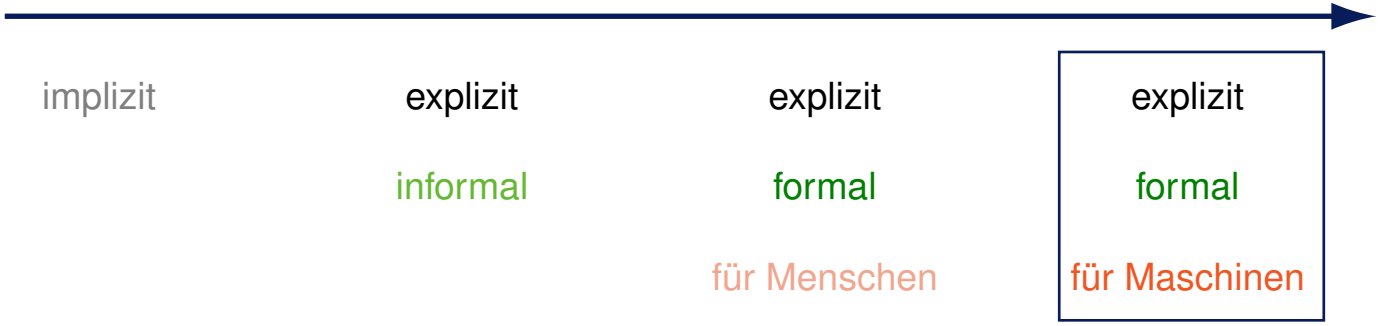


Formale Sprache mit expliziter Semantik:

- Beispiel: XML-Schema, Ontologie

Semantik im Web

„The Semantic Continuum“



Modelltheoretische Semantik, axiomatische Semantik:

- Schaffen die Voraussetzung, um das Wesen (die Semantik) von *Schlussfolgerungsprozessen* formal zu fassen

➜ Schlussfolgerungsprozesse von Mensch und Maschine sind nicht unterscheidbar

➜ die menschliche Semantik (des Schlussfolgerungsprozesses) ist auf Rechner übertragbar

- Beispiele: regelbasiertes System, Inferenz-System für Ontologie

Bemerkungen:

- Aristoteles entdeckte – in der strengen Satzform der Syllogismen – den Folgerungsbegriff als Teil unseres (westlichen) Denkens, der sich in der *natürlichen Sprache* widerspiegelt. Er begründete die Logik, vornehmlich als eine Theorie des Schlussfolgerns, an die sich folgerichtiges Denken und Argumentieren zu halten hat.
- Der Folgerungsbegriff für *formale Sprachen* wird in der Modelltheorie auf Basis von Interpretationen definiert, die Wahrheitsbedingungen für Ausdrücke der Sprache beschreiben. Dabei leitet sich die Wahrheit eines Satzes der formalen Sprache rekursiv aus der Wahrheit seiner Bestandteile ab. Stichwort: Tarski-Semantik
- Der modelltheoretische Folgerungsbegriff fordert einen Zusammenhang zwischen Prämissen und Konklusion, der unter allen Interpretationen Bestand hat.

  Dadurch wird der Folgerungsbegriff unabhängig von konkreten Interpretationen und deshalb automatisierbar.
- Eine Sprache, in der sich der Wahrheitswert ganzer Sätze auf Basis der Wahrheitwerte ihrer Symbole *in eindeutiger Weise* berechnen lässt, kann man zum Axiomatisieren (= Beschreiben der Welt) und zum automatischen Schlussfolgern benutzen. Einigen sich Mensch und Rechner auf eine solche Sprache, so sind ihre Schlussfolgerungen diesselben – andersherum: Der Rechner kann den Schlussfolgerungsprozess des Menschen nachbilden.

Semantik im Web

Schlussfolgerungsprozess auf Rechner übertragbar

Wissen

- Wissensrepräsentation
  - Semantische Netze
    - Begriffshierachien, Ontologien
      - Schlussfolgern über Ontologien
        - Prädikatenlogik, Frame-Logik, Beschreibungslogik
          - Kalküle, Inferenzsysteme, Regelverarbeitung
            - Korrektheit, Vollständigkeit, Entscheidbarkeit, Effizienz

Semantik von RDF/RDFS

Beispiel 1 (vgl. RDF/RDFS-Graph)

```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

Semantik von RDF/RDFS

Beispiel 1 (vgl. RDF/RDFS-Graph)

```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

Axiomatische Formulierung der Semantik:

$$\forall x \; \mathit{Type}(x, \text{lecturer}) \rightarrow \mathit{Type}(x, \text{teaching-staff})$$

$$\forall x \; \mathit{SubClassOf}(x, \text{lecturer}) \rightarrow \mathit{SubClassOf}(x, \text{teaching-staff})$$

Semantik von RDF/RDFS

Beispiel 1 (vgl. RDF/RDFS-Graph)

```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

Axiomatische Formulierung der Semantik:

$$\forall x \; \mathit{Type}(x, \text{lecturer}) \rightarrow \mathit{Type}(x, \text{teaching-staff})$$

$$\forall x \; \mathit{SubClassOf}(x, \text{lecturer}) \rightarrow \mathit{SubClassOf}(x, \text{teaching-staff})$$

```
<rdf:Description rdf:about="http://www.buw.de/Benno-Stein">
  <rdf:type rdf:resource="http://www.buw.de/lecturer"/>
</rdf:Description>
```

Formulierung als Grundprädikat:

$$\mathit{Type}(\text{Benno-Stein}, \text{lecturer})$$

Semantik von RDF/RDFS

Beispiel 1 (vgl. RDF/RDFS-Graph)

```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

Axiomatische Formulierung der Semantik:

$$\forall x \; \textit{Type}(x, \text{lecturer}) \rightarrow \textit{Type}(x, \text{teaching-staff})$$

$$\forall x \; \textit{SubClassOf}(x, \text{lecturer}) \rightarrow \textit{SubClassOf}(x, \text{teaching-staff})$$

```
<rdf:Description rdf:about="http://www.buw.de/Benno-Stein">
  <rdf:type rdf:resource="http://www.buw.de/lecturer"/>
</rdf:Description>
```

Formulierung als Grundprädikat:

$$\textit{Type}(\text{Benno-Stein}, \text{lecturer})$$

Auf Basis der RDF/RDF-Semantik schlussfolgerbar:

$$\textit{Type}(\text{Benno-Stein}, \text{teaching-staff})$$

Semantik von RDF/RDFS

Beispiel 2 (vgl. RDF/RDFS-Graph)

```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:domain rdf:resource="http://www.buw.de/course"/>
  <rdfs:range rdf:resource="http://www.buw.de/lecturer"/>
</rdfs:Property>
```

Semantik von RDF/RDFS

Beispiel 2 (vgl. RDF/RDFS-Graph)

```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:domain rdf:resource="http://www.buw.de/course"/>
  <rdfs:range rdf:resource="http://www.buw.de/lecturer"/>
</rdfs:Property>
```

Axiomatische Formulierung der Semantik:

$$\forall x \forall y \; \mathit{IsTaughtBy}(x, y) \rightarrow (\mathit{Type}(x, \text{course}) \wedge \mathit{Type}(y, \text{lecturer}))$$

Semantik von RDF/RDFS

Beispiel 2 (vgl. RDF/RDFS-Graph)

```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:domain rdf:resource="http://www.buw.de/course"/>
  <rdfs:range rdf:resource="http://www.buw.de/lecturer"/>
</rdfs:Property>
```

Axiomatische Formulierung der Semantik:

$$\forall x \forall y \; \mathit{IsTaughtBy}(x, y) \rightarrow (\mathit{Type}(x, \text{course}) \wedge \mathit{Type}(y, \text{lecturer}))$$

```
<rdf:Description rdf:about="http://www.buw.de/webTec-II">
  <buw:isTaughtBy rdf:resource="http://www.buw.de/Benno-Stein"/>
</rdf:Description>
```

Formulierung als Grundprädikat:

$$\mathit{IsTaughtBy}(\text{webTec-II}, \text{Benno-Stein})$$

Semantik von RDF/RDFS

Beispiel 2 (vgl. RDF/RDFS-Graph)

```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:domain rdf:resource="http://www.buw.de/course"/>
  <rdfs:range rdf:resource="http://www.buw.de/lecturer"/>
</rdfs:Property>
```

Axiomatische Formulierung der Semantik:

$$\forall x \forall y \; \mathit{IsTaughtBy}(x, y) \rightarrow (\mathit{Type}(x, \text{course}) \wedge \mathit{Type}(y, \text{lecturer}))$$

```
<rdf:Description rdf:about="http://www.buw.de/webTec-II">
  <buw:isTaughtBy rdf:resource="http://www.buw.de/Benno-Stein"/>
</rdf:Description>
```

Formulierung als Grundprädikat:

$$\mathit{IsTaughtBy}(\text{webTec-II}, \text{Benno-Stein})$$

Schlussfolgerbar:

$$\mathit{Type}(\text{webTec-II}, \text{course}), \; \mathit{Type}(\text{Benno-Stein}, \text{lecturer})$$

Semantik von RDF/RDFS

Axiomatische Formulierung der Semantik von RDFS

- `rdf:type, rdfs:subClassOf`

  $\forall x \forall y \forall z$ *Type*$(x, y) \wedge$ *SubClassOf*$(y, z) \rightarrow$ *Type*(x, z)

  $\forall x \forall y \forall z$ *SubClassOf*$(x, y) \wedge$ *SubClassOf*$(y, z) \rightarrow$ *SubClassOf*(x, z)

- `rdfs:subPropertyOf`

  $\forall x \forall y \forall z$ *SubPropertyOf*$(x, y) \wedge$ *SubPropertyOf*$(y, z) \rightarrow$ *SubPropertyOf*(x, z)

  $\forall x \forall y \forall z \forall v$ *Statement*$(x, y, z) \wedge$ *SubPropertyOf*$(y, v) \rightarrow$ *Statement*(x, v, z)

- `...`

Bemerkungen:

- Die hier vorgestellte Axiomatierung ist nicht vollständig; eine ausführlichere Darstellung kann in [Champin 2000] oder [Antoniou/Harmelen 2004] gefunden werden.
- Jede Axiomatisierung der Semantik von RDF/RDFS basiert auf der in http://www.w3.org/TR/rdf-mt/ vereinbarten modelltheoretischen Semantik.
- Auf Basis der axiomatischen Semantik von RDF/RDFS kann man mit Beweissystemen (Kalkülen) der Prädikatenlogik Folgerungen herleiten bzw. die Wahrheit von Formeln unter der Annahme der Wahrheit der Axiome beweisen. Da jedoch nur ein Teil der Mächtigkeit der Prädikatenlogik in der Axiomatisierung ausgenutzt wird, stellen entsprechende Kalküle einen „Overkill“ dar – zumal ihre Anwendung erhebliche Komplexitätsprobleme mit sich bringt.
- Eine weitere Möglichkeit zur Spezifikation der Semantik von RDF/RDFS besteht in der Angabe einer Menge spezieller Inferenzregeln, die auf RDF/RDFS zugeschnitten sind und für die die Vollständigkeit und Korrektheit bzgl. der Semantik von RDF/RDFS gezeigt wurde. Man könnte in diesem Zusammenhang von einem speziellen RDF/RDFS-Kalkül (Inferenzsystem) sprechen.

Semantik von RDF/RDFS

Inferenzregeln zur Operationalisierung der Semantik

Schema der Inferenz-Regeln:

`IF` E contains *<triple>* [and *<triple>*]
`THEN` add to E *<triple>*

Semantik von RDF/RDFS

Inferenzregeln zur Operationalisierung der Semantik

Schema der Inferenz-Regeln:

`IF` E contains *<triple>* [and *<triple>*]
`THEN` add to E *<triple>*

Beispiele:

`IF` E contains (x, p, y)
`THEN` add to E $(p,\ \texttt{rdf:type},\ \texttt{rdf:property})$

Semantik von RDF/RDFS

Inferenzregeln zur Operationalisierung der Semantik

Schema der Inferenz-Regeln:

`IF` E contains *<triple>* [and *<triple>*]
`THEN` add to E *<triple>*

Beispiele:

`IF` E contains (x, p, y)
`THEN` add to E $(p,\ \texttt{rdf:type},\ \texttt{rdf:property})$

`IF` E contains $(u,\ \texttt{rdfs:subClassOf}, v)$
and $(v,\ \texttt{rdfs:subClassOf}, w)$
`THEN` add to E $(u,\ \texttt{rdfs:subClassOf}, w)$

Semantik von RDF/RDFS

Inferenzregeln zur Operationalisierung der Semantik

Schema der Inferenz-Regeln:

`IF` E contains *<triple>* [and *<triple>*]
`THEN` add to E *<triple>*

Beispiele:

`IF` E contains (x, p, y)
`THEN` add to E $(p,\ \texttt{rdf:type},\ \texttt{rdf:property})$

`IF` E contains $(u,\ \texttt{rdfs:subClassOf}, v)$
and $(v,\ \texttt{rdfs:subClassOf}, w)$
`THEN` add to E $(u,\ \texttt{rdfs:subClassOf}, w)$

`IF` E contains $(x,\ \texttt{rdfs:type}, u)$
and $(u,\ \texttt{rdfs:subClassOf}, v)$
`THEN` add to E $(x,\ \texttt{rdf:type}, v)$

Bemerkungen:

- E bezeichnet eine Menge beliebiger RDF-Tripel.
- Eine Menge von Inferenzregeln bezeichnet man als Kalkül.

Semantik von RDF/RDFS

Grundlagen der Logik

Die Semantik von Schlussfolgerungsprozessen kann durch Rechner nachgebildet werden. In diesem Zusammenhang spielen insbesondere folgende Begriffe der Logik eine Rolle:

1. *Objektebene.* Syntax / Grammatik einer formalen Sprache
2. *Objektebene.* Semantik / Interpretation einer formalen Sprache
3. *Metaebene.* Schlussfolgern über Sätze eine formale Sprache

Mechanisierung bzw. Automatisierung des Schlussfolgerns:

- Kalkül
- Korrektheit und Vollständigkeit eines Kalküls
- Entscheidbarkeit

Semantik von RDF/RDFS

Grundlagen der Logik

Aufbau der Vorlesung `Logik` [Lettmann/Stein]:

Aussagenlogik

- Syntax
- Semantik
- Folgerungsbegriff
- Formeltransformation
- Normalformen
- Erfüllbarkeitsalgorithmen
- semantische Bäume
- syntaktische Schlußfolgerungsverfahren
- Erfüllbarkeitsprobleme

Prädikatenlogik

- Syntax
- Semantik
- Äquivalenzen
- Normalformen
- Standarderfüllbarkeit
- prädikatenlogische Resolution

Semantik von RDF/RDFS

Quellen zum Nachlernen und Nachschlagen im Web

- E. Franconi. Tutorien für Aussagen-, Prädikaten- und Beschreibungslogik.
  http://www.inf.unibz.it/ franconi/dl/course/
- P. Hayes, Ed. *RDF Semantics.*
  W3C Recommendation. http://www.w3.org/TR/rdf-mt/

Kapitel WT: VIII (Fortsetzung)

VIII. Semantic Web

- WWW heute
- Semantic Web Vision
- RDF: Einführung
- RDF: Konzepte
- RDF: XML-Serialisierung
- RDF: Anwendungen
- RDFS: Einführung
- RDFS: Konzepte
- Semantik im Web
- Semantik von RDF/RDFS
- Ontologien
- OWL: Konzepte
- OWL: Logikhintergrund
- OWL: Anwendungen

Ontologien

Definition

Ontologie [griechisch] – die Lehre vom Seienden

Ontologien

Definition

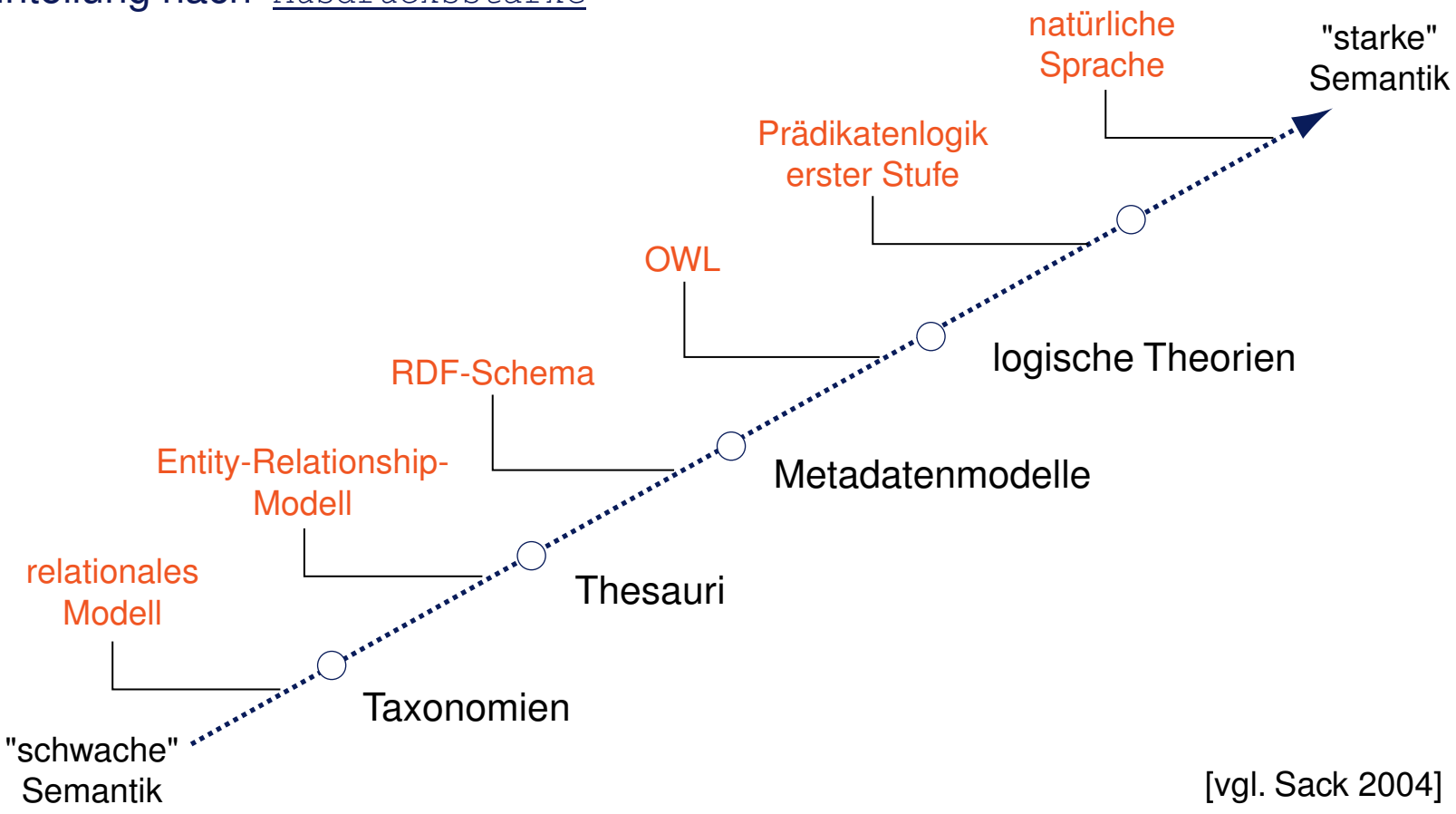
Ontologie [griechisch] – die Lehre vom Seienden

Definition 1 (Ontology [Gruber 1993]**)**
An ontology is a formal specification of a shared conceptualization of a domain of interest.

formal specification	➜	interpretierbar durch Maschinen
of a shared	➜	gemeinsames Verständnis einer Gruppe
conceptualization	➜	es geht um Begriffsbildung
of a domain of interest	➜	bekannter (abgegrenzter) Gegenstandsbereich

Ontologien

Einteilung nach Ausdrucksstärke



[vgl. Sack 2004]

Je reicher die Möglichkeiten zur Ontologiebeschreibung, um so stärker ist die intendierte Semantik, und um so komplexer können Schlussfolgerungsprozesse im Worst-Case werden.

Ontologien

Taxonomien

Um 1740 entwickelt Carl von Lineé ein hierarchisches Klassifikationsschema für Tiere und Pflanzen.

Systematik der Silbermöwe:

Reich (Regnum)	Tiere
Unterreich (Subregnum)	Vielzeller Metazoa
Abteilung (Divisio)	Vielzeller im engeren Sinne Eumetazoa
Stamm (Phylum)	Chordatiere Chordata
Unterstamm (Subphylum)	Wirbeltiere Vertebrata
Klasse (Classis)	Vögel Aves
Ordnung (Ordo)	Wat- und Möwenvögel Charadriiformes
Unterordnung (Subordo)	Möwenartige Lari
Familie (Familia)	Möwenvögel Laridae
Unterfamilie (Subfamilia)	Larinae
Gattung (Genus)	Möwen Larus
Art (Species)	Silbermöwe Larus argentatus

Ontologien

Sinn und Zweck

Ontologien ermöglichen:

- Realisierung „semantischer“ Suchfunktionen
  Beispiele: automatische Spezialisierung, Generalisierung, Verfeinerung
- Realisierung „intelligenter“ Dialoge mit Maschinen
  Beispiel: Nachfragen bei Mehrdeutigkeiten
- Interoperabilität beim Austausch Daten, die ein Markup besitzen

Ontologien

Sinn und Zweck

Ontologien ermöglichen:

- Realisierung „semantischer“ Suchfunktionen
  Beispiele: automatische Spezialisierung, Generalisierung, Verfeinerung
- Realisierung „intelligenter“ Dialoge mit Maschinen
  Beispiel: Nachfragen bei Mehrdeutigkeiten
- Interoperabilität beim Austausch Daten, die ein Markup besitzen

Elemente einer Ontologie-Sprache:

- Klassen, auch Konzepte genannt
- Eigenschaften der Klassen
- (binäre) Beziehungen zwischen Klassen und Eigenschaften

Anwendungen von formal beschriebenen Ontologien basieren darauf, logische Schlussfolgerungen über (Teil)Mengen von Konzepten zu ziehen.

Ontologien

Grenzen von RDF-Schema

1. RDF-Schema ist eine einfache Ontologie-Sprache.
2. „Echte“ Ontologie-Sprachen stellen ein reicheres Beschreibungsvokabular als RDF-Schema zur Verfügung.

Einige Defizite von RDF-Schema:

- Mit `rdfs:subClassOf` kann keine Disjunktheit ausgedrückt werden.
- Die Individuen aller Unterklassen einer Klasse können nicht in ihrer Gesamtheit angesprochen werden.
- Kardinalitätsangaben für Properties sind nicht formulierbar.
- Spezielle Eigenschaften von Properties wie Transitivität, Umkehrbarkeit oder Funktionscharakter sind nicht formulierbar.

OWL: Konzepte

Bemerkungen:

- Das Akronym für Web Ontology Language hätte eigentlich WOL, nicht OWL sein müssen. In Referenz auf die literarische Figur der Eule aus Milnes „Pu der Bär“, die als einziges Tier im Wald ihren Namen schreiben konnte (und denselben aber mit einem Buchstabendreher im englischen Original WOL statt OWL schreibt), wurde dieser Buchstabendreher umgekehrt für die OWL übernommen.

  [de.wikipedia.org/wiki/Web_Ontology_Language]

OWL: Konzepte

Historie von Ontologie-Sprachen für das Web

1995 SHOE. Simple HTML Ontology Extensions.

1997 OIL. Ontology Inference Layer.
Die erste Ontologiesprache basierend auf RDF und XML-Schema.

2000 DAML. DARPA Agent Markup Language.

2001 DAML+OIL. Eine „Joint Ontology Language“ zwischen Europa und USA.
Konzepte dieser Sprache haben große Ähnlichkeit mit Konzepten in RDFS.

2001 Gründung der Ontology Working Group WebOnt des W3C.

2002 Erstes Release der „OWL Use Cases and Requirements“.

2004 OWL Web Ontology Language. Semantik und abstrakte Syntax.
Recommendation.

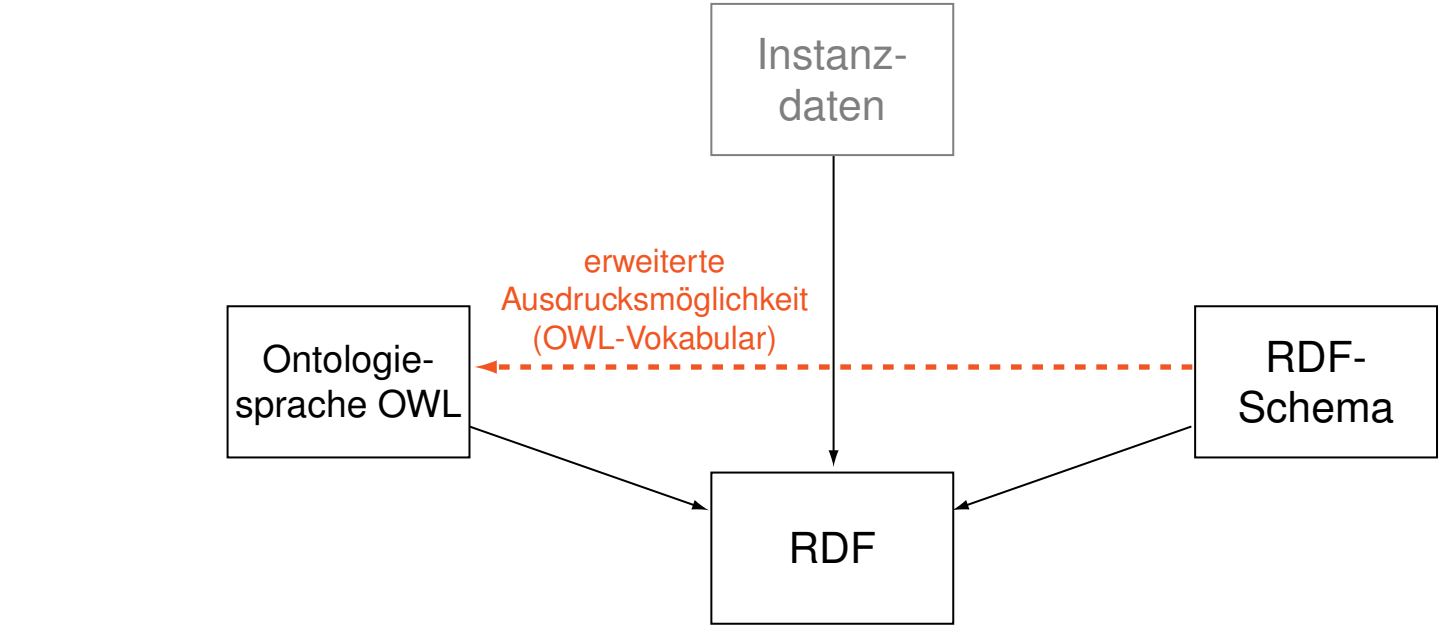
OWL: Konzepte

Zusammenhang zwischen OWL und RDF/RDFS

[vgl. Boley 2001]

OWL: Konzepte

Zusammenhang zwischen OWL und RDF/RDFS



[vgl. Boley 2001]

Bemerkungen:

- An OWL ontology is an RDF graph, which is in turn a set of RDF triples. As with any RDF graph, an OWL ontology graph can be written in many different syntactic forms.

  [www.w3.org/TR/owl-ref/]

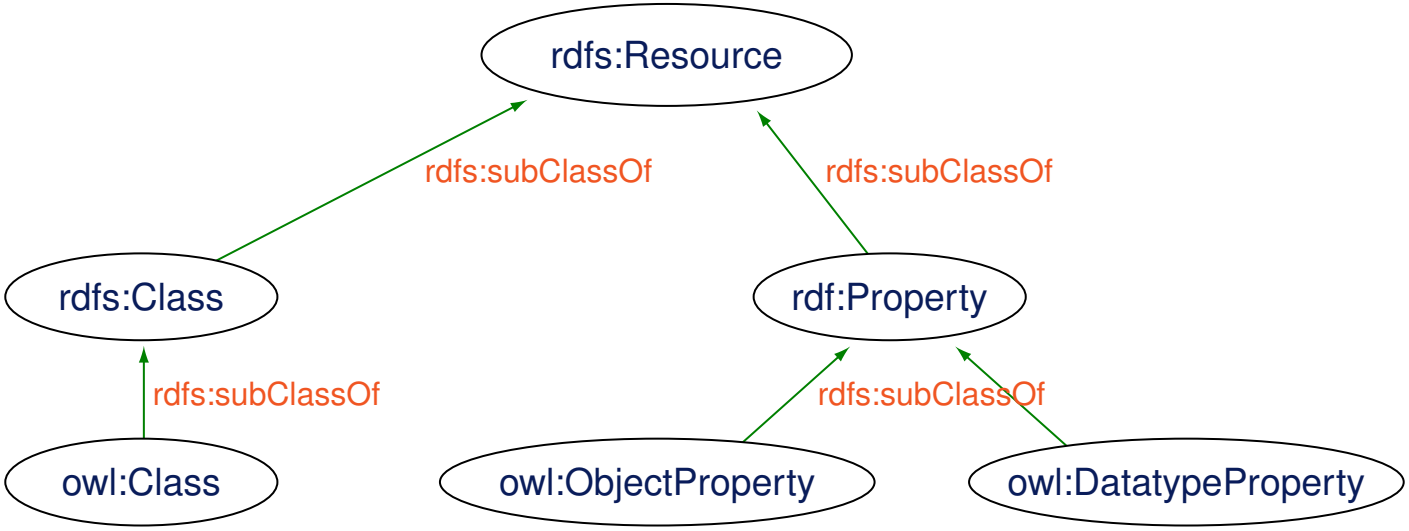
- Zu den Design-Zielen von OWL gehört:
  - vernünftiger Kompromiss zwischen Ausdrucksfähigkeit und inferentieller Komplexität
  - Skalierbarkeit – insbesondere vor dem Hintergrund des World Wide Web
  - Kompatibilität zu existierenden Standards des WWW

OWL: Konzepte

Zusammenhang zwischen OWL und RDF/RDFS (Fortsetzung)

Eine OWL-Ontologie definiert Klassen, Individuen und Properties.

Individuen = Instanzen von Klassen



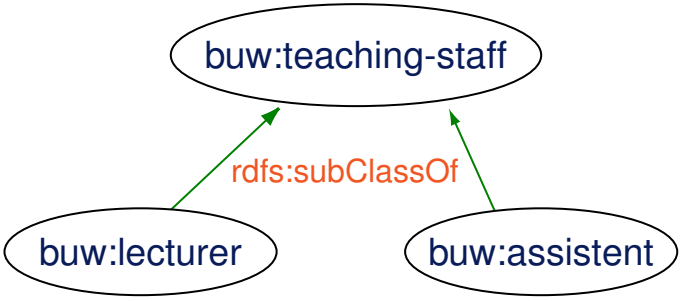
Unterscheidung von zwei Arten von Properties:

1. Object-Properties, um Beziehungen zwischen Individuen auszudrücken.
2. Datentyp-Properties, um Individuen mit Datentypen zu assoziieren.

OWL: Konzepte

Klassen

Definition von Klassen im Prinzip wie in RDFS:

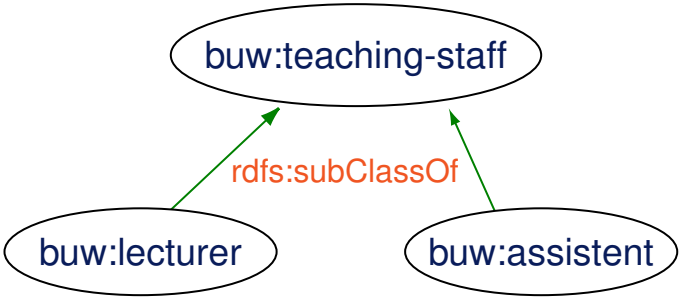



```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>
```

OWL: Konzepte

Klassen

Definition von Klassen im Prinzip wie in RDFS:



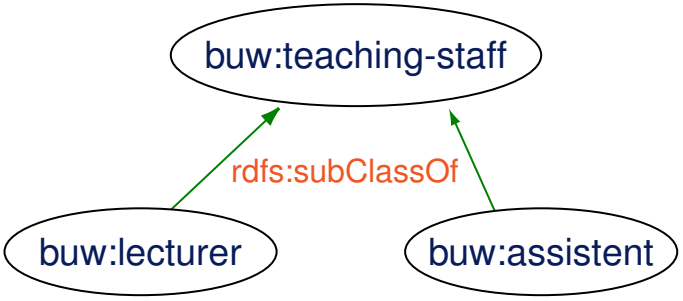

```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>

<owl:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</owl:Class>
```

OWL: Konzepte

Klassen

Definition von Klassen im Prinzip wie in RDFS:




```
<rdfs:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</rdfs:Class>

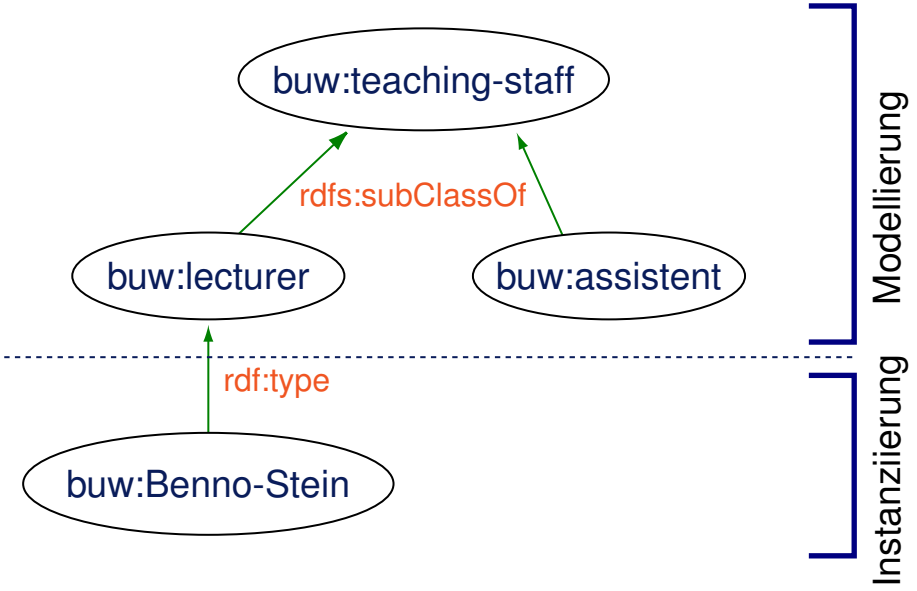
<owl:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</owl:Class>

<owl:Class rdf:about="http://www.buw.de/lecturer">
  <owl:disjointWith rdf:resource="http://www.buw.de/assistent"/>
</owl:Class>
```

OWL: Konzepte

Klassen

Instanzen von Klassen (Individuen) werden wie in RDF deklariert.




```
<rdf:Description rdf:about="http://www.buw.de/Benno-Stein">
  <rdf:type rdf:resource="http://www.buw.de/lecturer"/>
</rdf:Description>
```

bzw.

```
<buw:lecturer rdf:about="http://www.buw.de/Benno-Stein"/>
```

OWL: Konzepte

Klassen: Deklaration von Verknüpfungen und Restriktionen

- Vereinigung: `owl:unionOf`
- Durchschnittbildung: `owl:intersectionOf`
- Komplementbildung: `owl:complementOf`
- Disjunktheitsforderung: `owl:disjointWith`
- Definition durch Aufzählung: `owl:oneOf`
- Deklaration als Äquivalenzklasse: `owl:equivalentClass`

OWL besitzt zwei vordefinierte Klassen:

1. Jede Klasse ist Unterklasse der allgemeinsten Klasse `owl:Thing`.
2. Jeder Klasse ist Oberklasse der leeren Klasse `owl:Nothing`.

OWL: Konzepte

Klassen: Beispiele [vgl. Sack 2004]

```
<owl:Class rdf:about="#Frucht">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SuessFrucht"/>
    <owl:Class rdf:about="#NichtSuessFrucht"/>
  </owl:unionOf>
</owl:Class>
```

OWL: Konzepte

Klassen: Beispiele [vgl. Sack 2004]

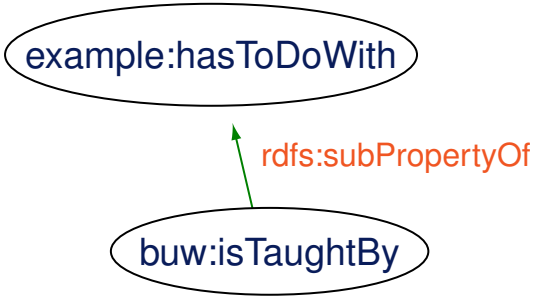
```
<owl:Class rdf:about="#Frucht">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SuessFrucht"/>
    <owl:Class rdf:about="#NichtSuessFrucht"/>
  </owl:unionOf>
</owl:Class>


<owl:Class rdf:ID="WeinFarbe">
  <rdfs:subClassOf rdf:resource="#WeinDeskriptor">
    <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#Weiss"/>
      <owl:Thing rdf:about="#Rosé"/>
      <owl:Thing rdf:about="#Rot"/>
    </owl:oneOf>
  </rdfs:subClassOf>
</owl:Class>
```

OWL: Konzepte

Object-Properties

Definition von Object-Properties im Prinzip wie in RDFS:

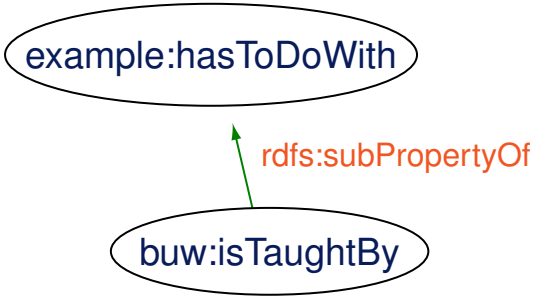



```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
</rdfs:Property>
```

OWL: Konzepte

Object-Properties

Definition von Object-Properties im Prinzip wie in RDFS:



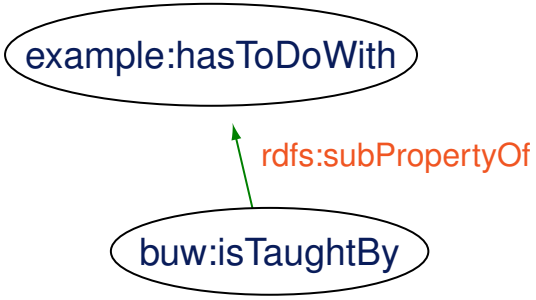

```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
</rdfs:Property>

<owl:ObjectProperty rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
</owl:ObjectProperty>
```

OWL: Konzepte

Object-Properties

Definition von Object-Properties im Prinzip wie in RDFS:




```
<rdfs:Property rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
</rdfs:Property>

<owl:ObjectProperty rdf:about="http://www.buw.de/isTaughtBy">
  <rdfs:subPropertyOf rdf:resource="&example;hasToDoWith"/>
</owl:ObjectProperty>
```

1. Properties können durch `rdfs:subPropertyOf` spezialisiert werden.
2. Für Properties können besondere Eigenschaften deklariert werden.
3. Domain und Range von Properties können eingeschränkt werden.

OWL: Konzepte

Object-Properties: Deklaration besonderer Eigenschaften *(Characteristics)*

- Transitivität: `owl:TransitiveProperty`.
  $P(x, y) \wedge P(y, z) \rightarrow P(x, z)$
- Symmetrie: `owl:SymmetricProperty`.
  $P(x, y) \leftrightarrow P(y, x)$

OWL: Konzepte

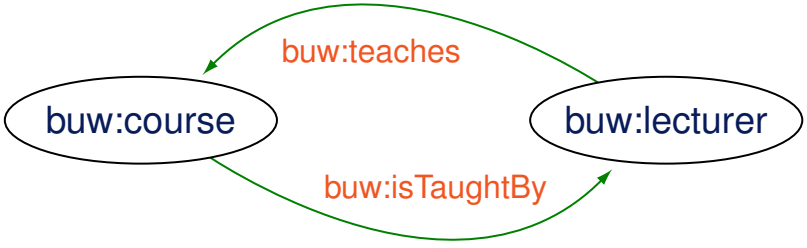
Object-Properties: Deklaration besonderer Eigenschaften *(Characteristics)*

- Transitivität: `owl:TransitiveProperty`.
  $P(x, y) \wedge P(y, z) \rightarrow P(x, z)$
- Symmetrie: `owl:SymmetricProperty`.
  $P(x, y) \leftrightarrow P(y, x)$
- Funktionscharakter: `owl:FunctionalProperty`.
  $P(x, y) \wedge P(x, z) \rightarrow \mathit{Equal}(y, z)$
- Umgekehrter Funktionscharakter: `owl:InverseFunctionalProperty`.
  $P(x, y) \wedge P(z, y) \rightarrow \mathit{Equal}(x, z)$

OWL: Konzepte

Object-Properties: Deklaration besonderer Eigenschaften *(Characteristics)*

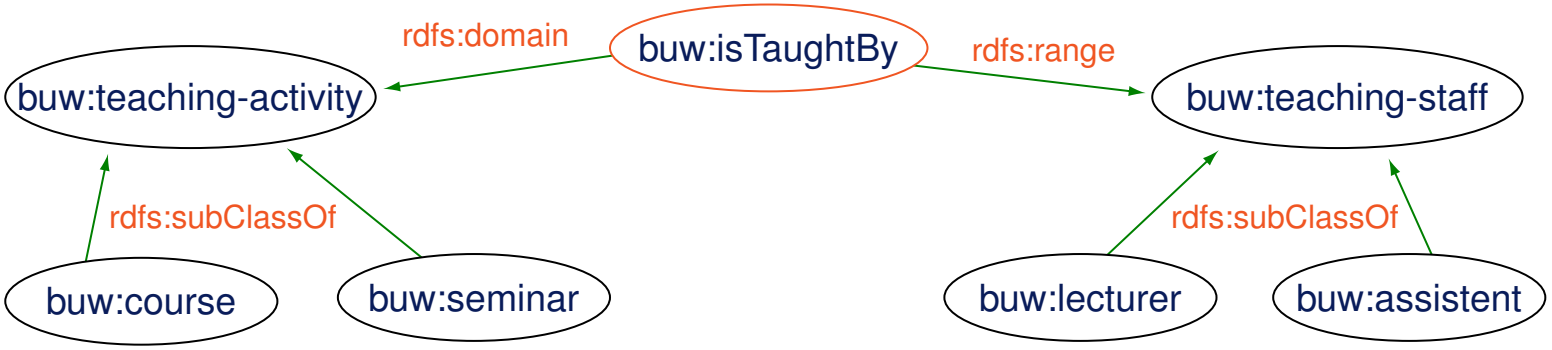
- Transitivität: `owl:TransitiveProperty`.
  $P(x,y) \wedge P(y,z) \rightarrow P(x,z)$
- Symmetrie: `owl:SymmetricProperty`.
  $P(x,y) \leftrightarrow P(y,x)$
- Funktionscharakter: `owl:FunctionalProperty`.
  $P(x,y) \wedge P(x,z) \rightarrow \mathit{Equal}(y,z)$
- Umgekehrter Funktionscharakter: `owl:InverseFunctionalProperty`.
  $P(x,y) \wedge P(z,y) \rightarrow \mathit{Equal}(x,z)$
- Umkehrbeziehung: `owl:inverseOf`.



OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen I

Zugrundeliegendes Schema, Theorie oder Ontologie:



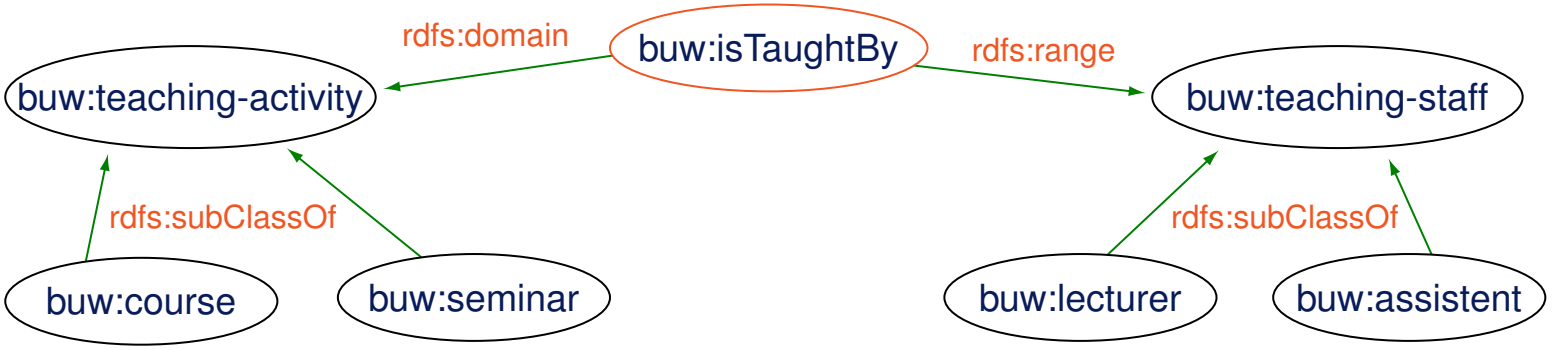
„Vorlesungen, die nur von Dozenten unterrichtet werden.“ =

„Die Elemente x *aus* `buw:course`*, denen bzgl. der Property* `buw:isTaughtBy` *nur Range-Elemente* y *aus* `buw:lecturer` *zugeordnet sind.“*

OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen I

Zugrundeliegendes Schema, Theorie oder Ontologie:



„Vorlesungen, die nur von Dozenten unterrichtet werden.“ =

„Die Elemente x *aus* `buw:course`*, denen bzgl. der Property* `buw:isTaughtBy` *nur Range-Elemente* y *aus* `buw:lecturer` *zugeordnet sind.“*

Formulierung in Prädikatenlogik:

$$\mathit{Course}(x) \wedge (\forall y : \mathit{IsTaughtBy}(x, y) \rightarrow \mathit{Lecturer}(y))$$

OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen I

Formulierung in RDF/XML:

```
<owl:Class rdf:about="http://www.buw.de/firstYearCourse">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/course"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.buw.de/isTaughtBy"/>
      <owl:allValuesFrom rdf:resource="http://www.buw.de/lecturer"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen I

Formulierung in RDF/XML:

```
<owl:Class rdf:about="http://www.buw.de/firstYearCourse">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/course"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.buw.de/isTaughtBy"/>
      <owl:allValuesFrom rdf:resource="http://www.buw.de/lecturer"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Formulierung in Prädikatenlogik (hier als Menge mit qualifizierender Formel) :

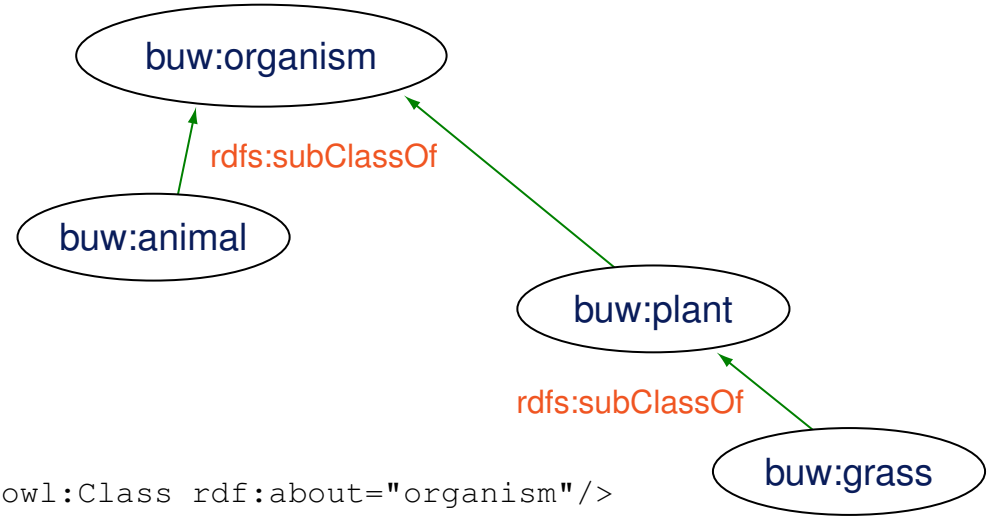
$$\{x \mid \mathit{Course}(x) \wedge (\forall y : \mathit{IsTaughtBy}(x, y) \rightarrow \mathit{Lecturer}(y))\}$$

Formulierung in Beschreibungslogik:

$$\mathrm{Course} \sqcap \forall \mathrm{IsTaughtBy}.\mathrm{Lecturer}$$

OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen II




```
<owl:Class rdf:about="organism"/>

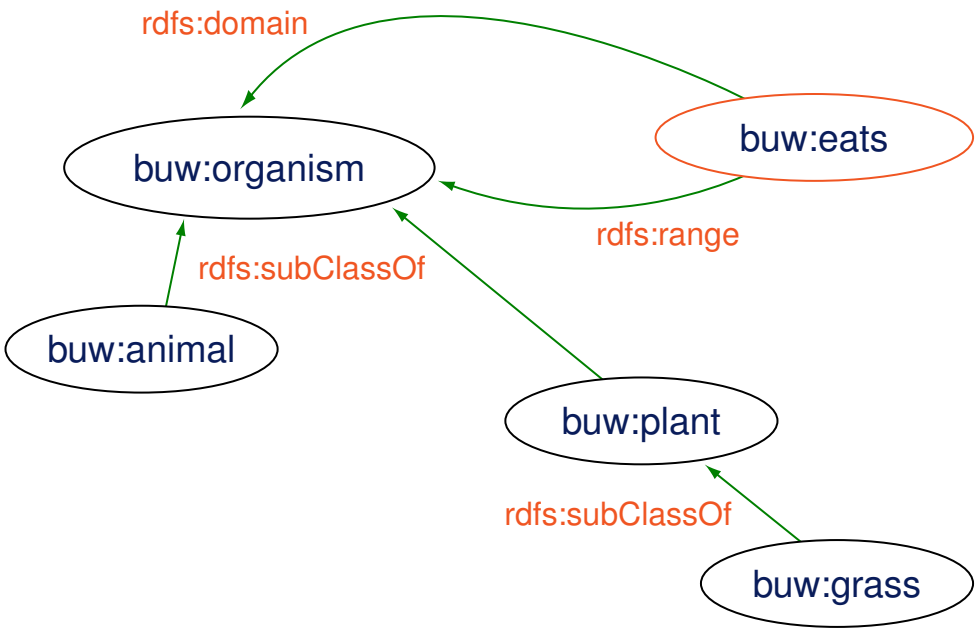
<owl:Class rdf:about="plant">
  <rdfs:subClassOf rdf:resource="organism"/>
  <owl:disjointWith rdf:resource="animal"/>
</owl:Class>

<owl:Class rdf:about="grass">
  <rdfs:subClassOf rdf:resource="plant"/>
</owl:Class>

<owl:Class rdf:about="animal">
  <rdfs:subClassOf rdf:resource="organism"/>
  <owl:disjointWith rdf:resource="plant"/>
</owl:Class>
```

OWL: Konzepte

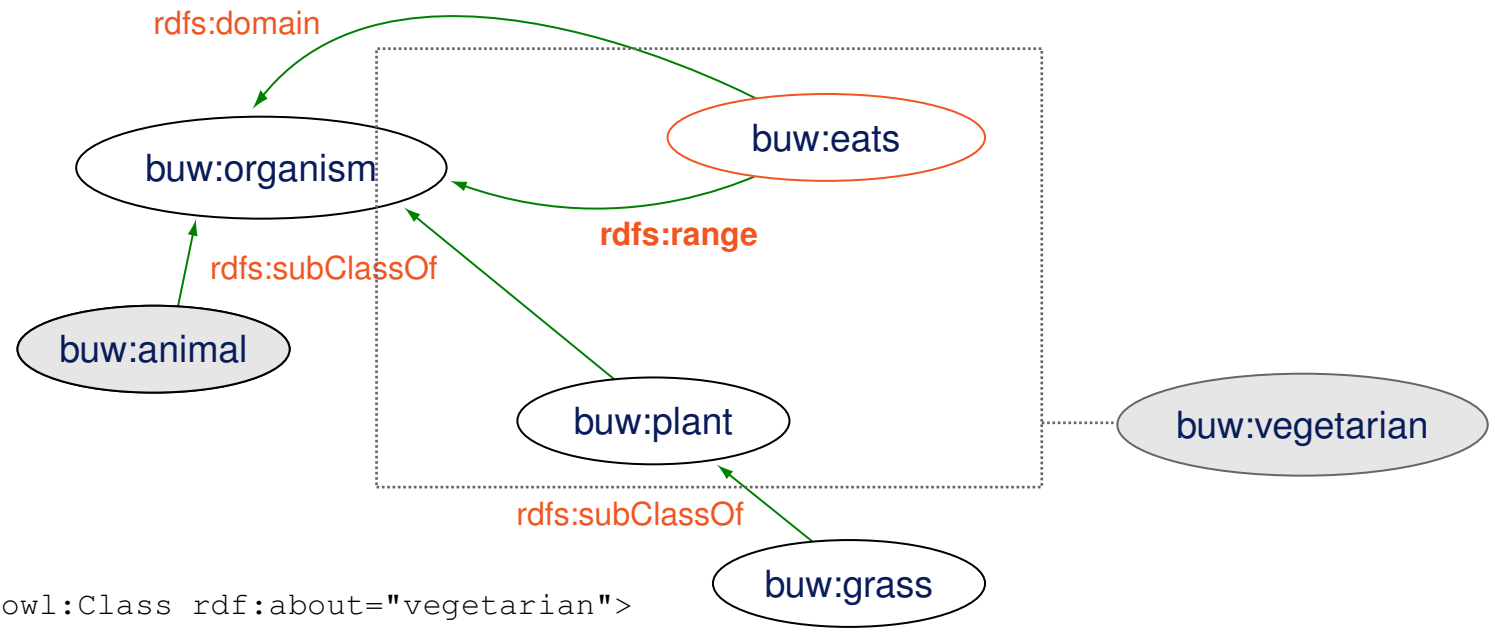
Range-Einschränkung: Konstruktion spezialisierter Klassen II




```
<owl:ObjectProperty rdf:about="eats">
  <rdfs:domain rdf:resource="organism" />
  <rdfs:range rdf:resource="organism" />
</owl:ObjectProperty>
```

OWL: Konzepte

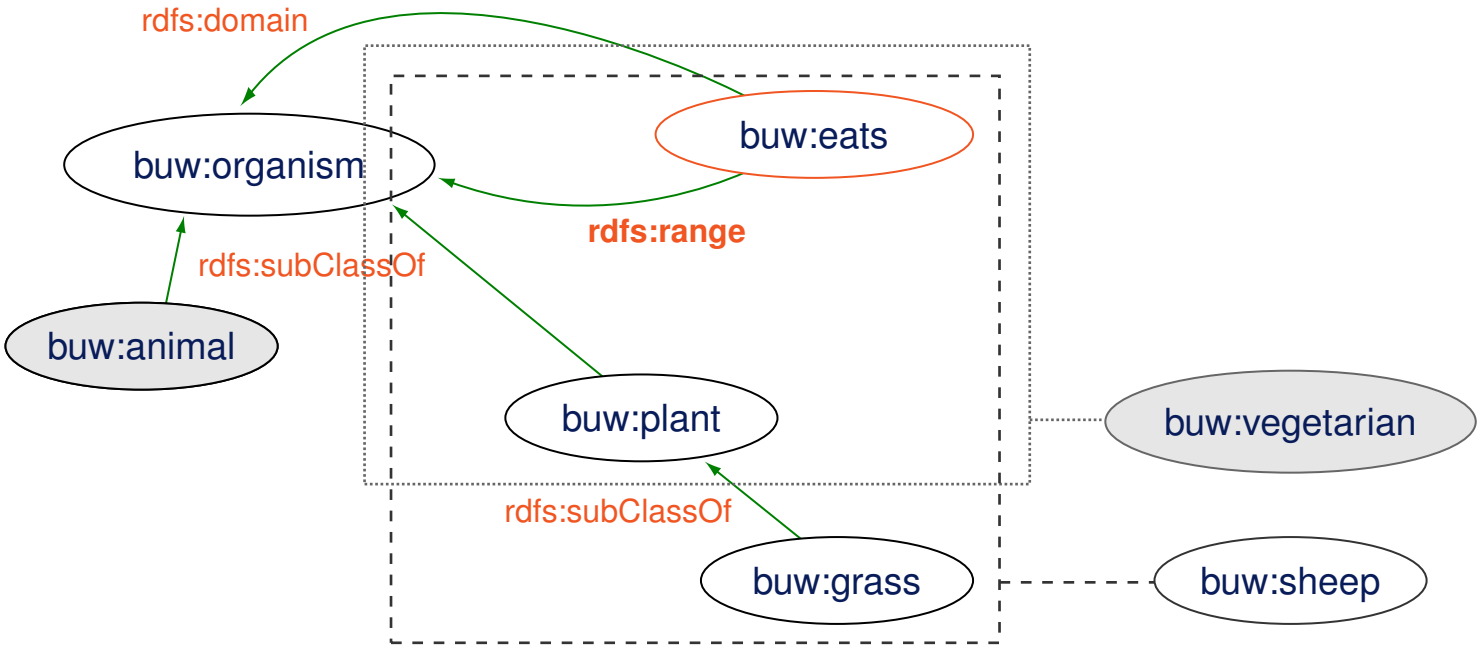
Range-Einschränkung: Konstruktion spezialisierter Klassen II




```
<owl:Class rdf:about="vegetarian">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="animal"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="eats"/>
          <owl:allValuesFrom rdf:resource="plant"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

OWL: Konzepte

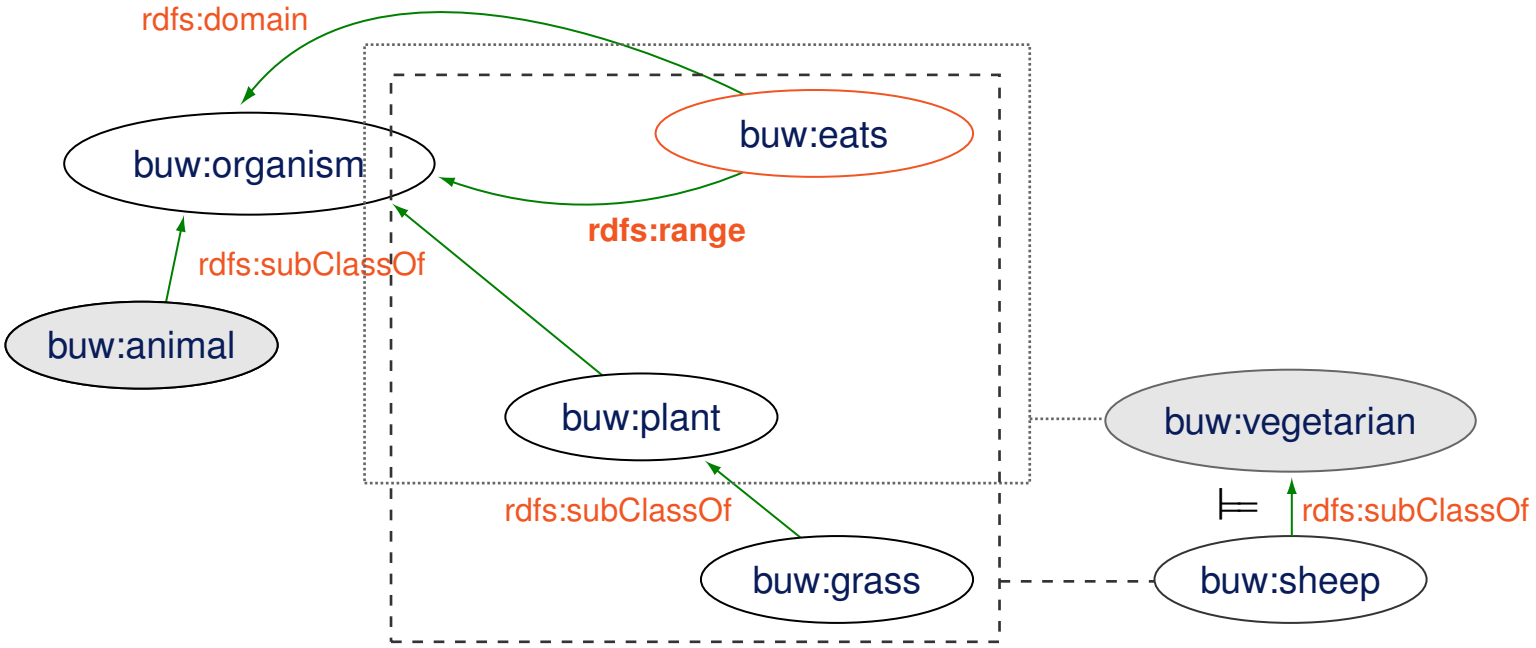
Range-Einschränkung: Konstruktion spezialisierter Klassen II




```
<owl:Class rdf:about="sheep">
  <rdfs:subClassOf rdf:resource="animal"/>
    <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="eats"/>
      <owl:allValuesFrom rdf:resource="grass"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL: Konzepte

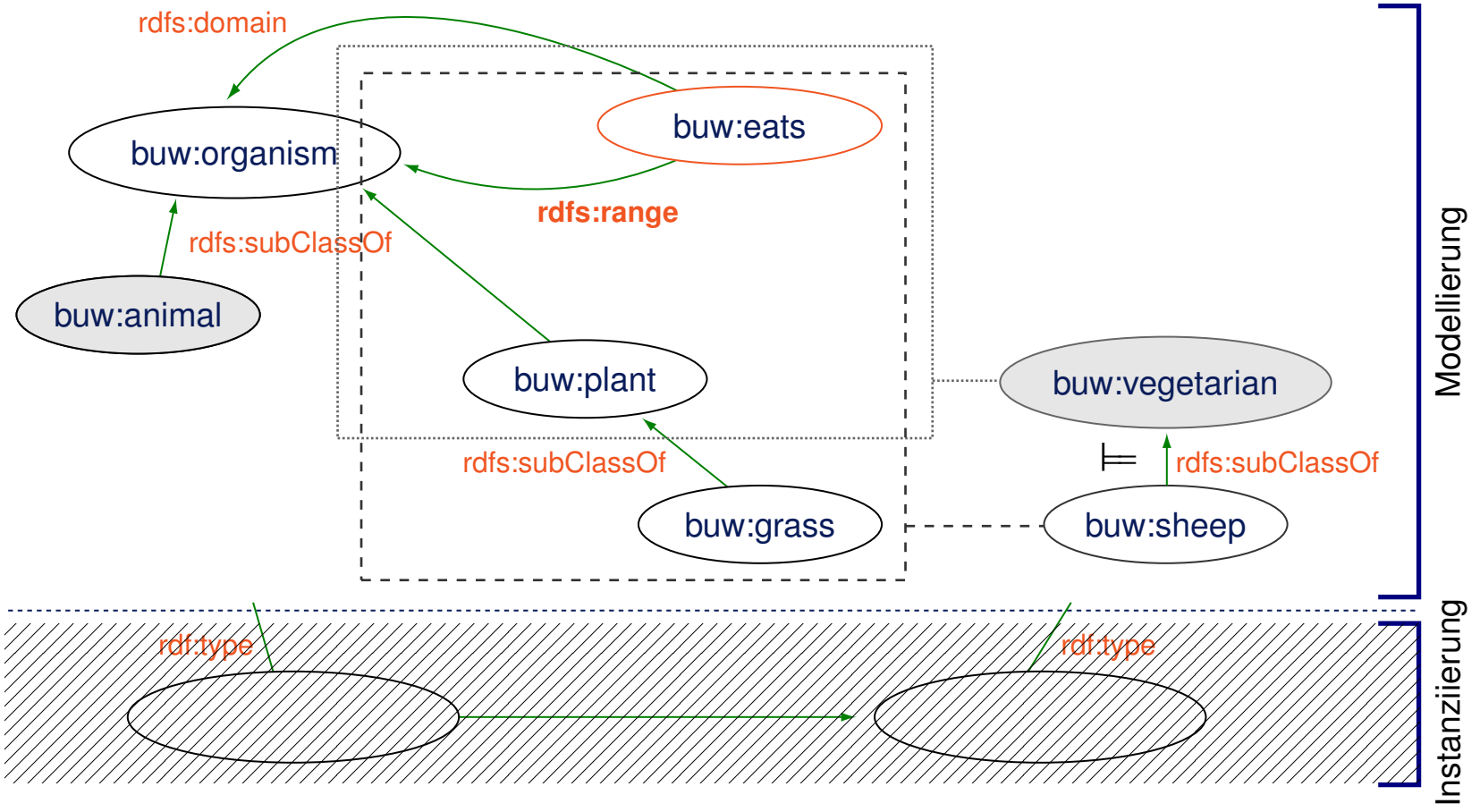
Range-Einschränkung: Konstruktion spezialisierter Klassen II



Schlussfolgerung auf der Modellierungsebene.

OWL: Konzepte

Range-Einschränkung: Konstruktion spezialisierter Klassen II



Bemerkungen:

- Wegen der Open-World-Semantic des Semantic Web kann jeder Anwender Instanzen für die Property buw:eats hinzufügen. Bei den hier konstruierten, spezialisierten Klassen (buw:vegetarian, buw:sheep) kann das zur Folge haben, dass Instanzen (Tiere), die zu einem Zeitpunkt t_1 zur Klasse buw:vegetarian oder buw:sheep gehörten, zu einem späteren Zeitpunkt t_2 nicht mehr dazu gehören, weil sie die `owl:allValuesFrom`-Restriktion nicht mehr erfüllen.

  Es handelt sich um eine nicht-monotone Situation: obwohl das Wissen (= Anzahl der Property-Instanzen) zunimmt, sinkt die Anzahl ableitbarer Schlussfolgerungen (= Anzahl der Instanzen in einer spezialisierten Klasse).
- Mit monotonen Schlussfolgerungsverfahren/-werkzeugen (Protégé, OWL-API) lassen sich nur Schlussfolgerungen ziehen, die für immer gültig sind. Die Klassenmitgliedschaft (Type-Property) bei den spezialisierten Klassen buw:vegetarian oder buw:sheep fällt nicht hierunter.

  Ein nicht-monotoner Reasoner kann mit dieser Siutation umgehen, in dem er gemachte Schlussfolgerungen wieder zurückzieht, falls sie nicht weiter gültig sind. Siehe Vorlesung `Logik` [Stein/Lettmann 1996-2008]: Nicht-monotones Schließen.
- Auf der Modellierungsebene ist die Problematik der Nicht-Monotonie nicht gegeben, weil die Ontologie vorliegt und über Eigenschaften ihrer Klassen zeitunabhängig geschlussfolgert werden kann.
- Es existieren weitere Möglichkeiten, um Restriktionen zu spezifizieren:
  - `owl:someValuesFrom`
  - `owl:hasValue`
  - `owl:cardinality, owl:minCardinality, owl:maxCardinality`

OWL: Konzepte

Header- und Metadaten für OWL-Ontologien

OWL-Dateien sind RDF-Dokumente und werden als OWL-Ontologien bezeichnet.

Üblicher RDF-Dokument-Header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#">
```

OWL: Konzepte

Header- und Metadaten für OWL-Ontologien

OWL-Dateien sind RDF-Dokumente und werden als OWL-Ontologien bezeichnet.

Üblicher RDF-Dokument-Header:

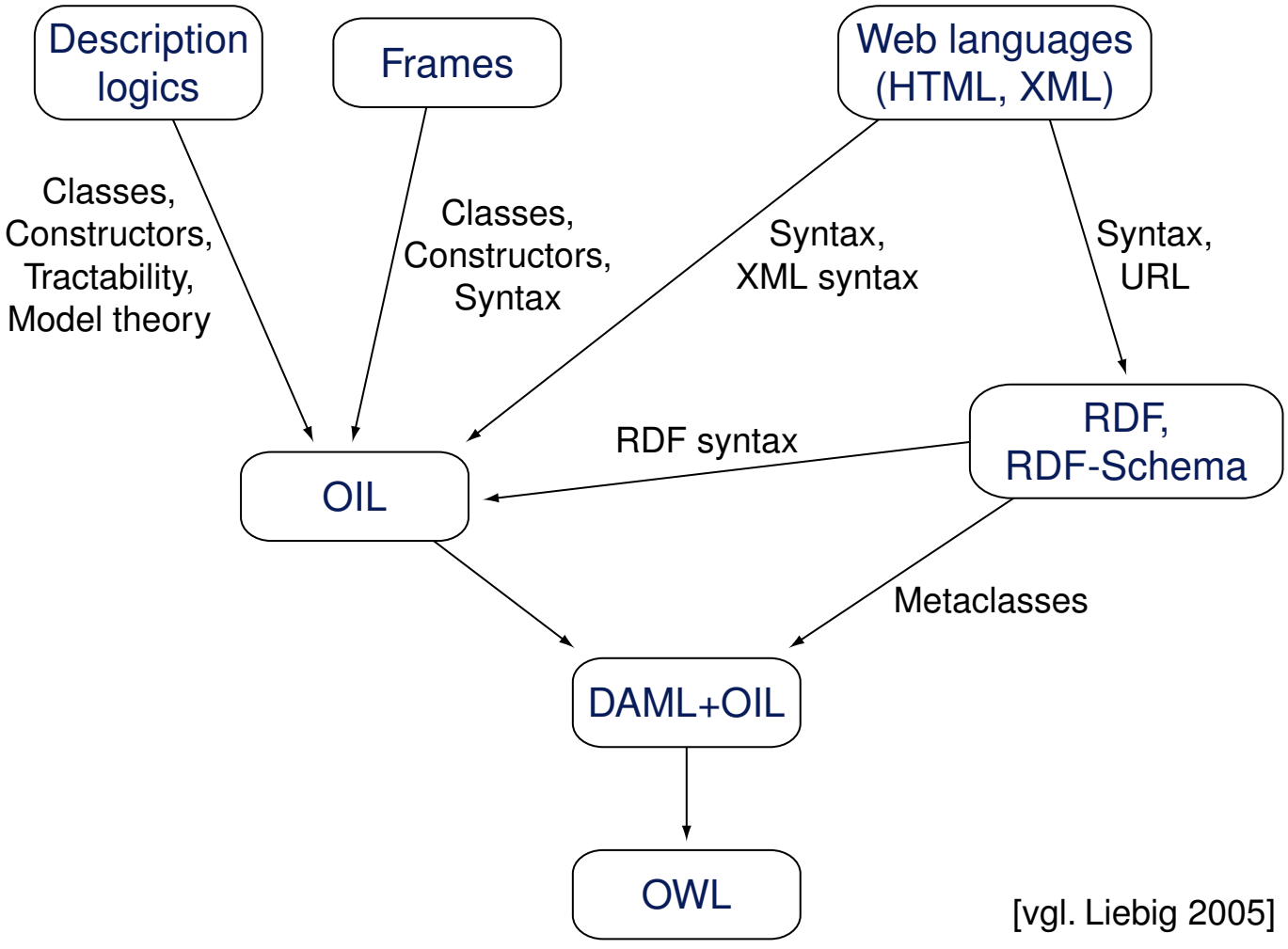
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#">
```

Ontologie-Metadaten:

```
  <owl:Ontology rdf:about="">
    <rdfs:comment>An example OWL ontology</rdfs:comment>
    <owl:priorVersion
       rdf:resource="http://buw-ontologies-old/teaching-ontology"/>
    <owl:imports rdf:resource="http://buw-ontologies/person-ontology"/>
    <rdfs:label>Teaching Ontology</rdfs:label>
  </owl:Ontology>
```

OWL: Logikhintergrund

Sprachliche Wurzeln von OWL



[vgl. Liebig 2005]

OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten

OWL lässt sich in der Prädikatenlogik (erster Stufe, PL1) axiomatisieren.

Warum sollte man diese nicht zur Beschreibung von Ontologien einsetzen?

OWL: Logikhintergrund

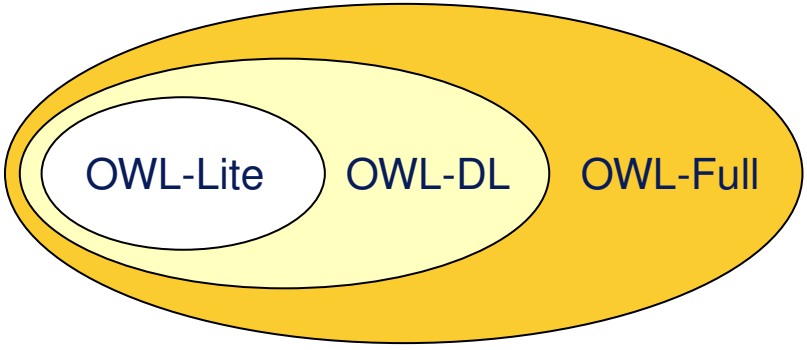
Ausdruckstärke von OWL-Varianten

OWL lässt sich in der Prädikatenlogik (erster Stufe, PL1) axiomatisieren.

Warum sollte man diese nicht zur Beschreibung von Ontologien einsetzen?

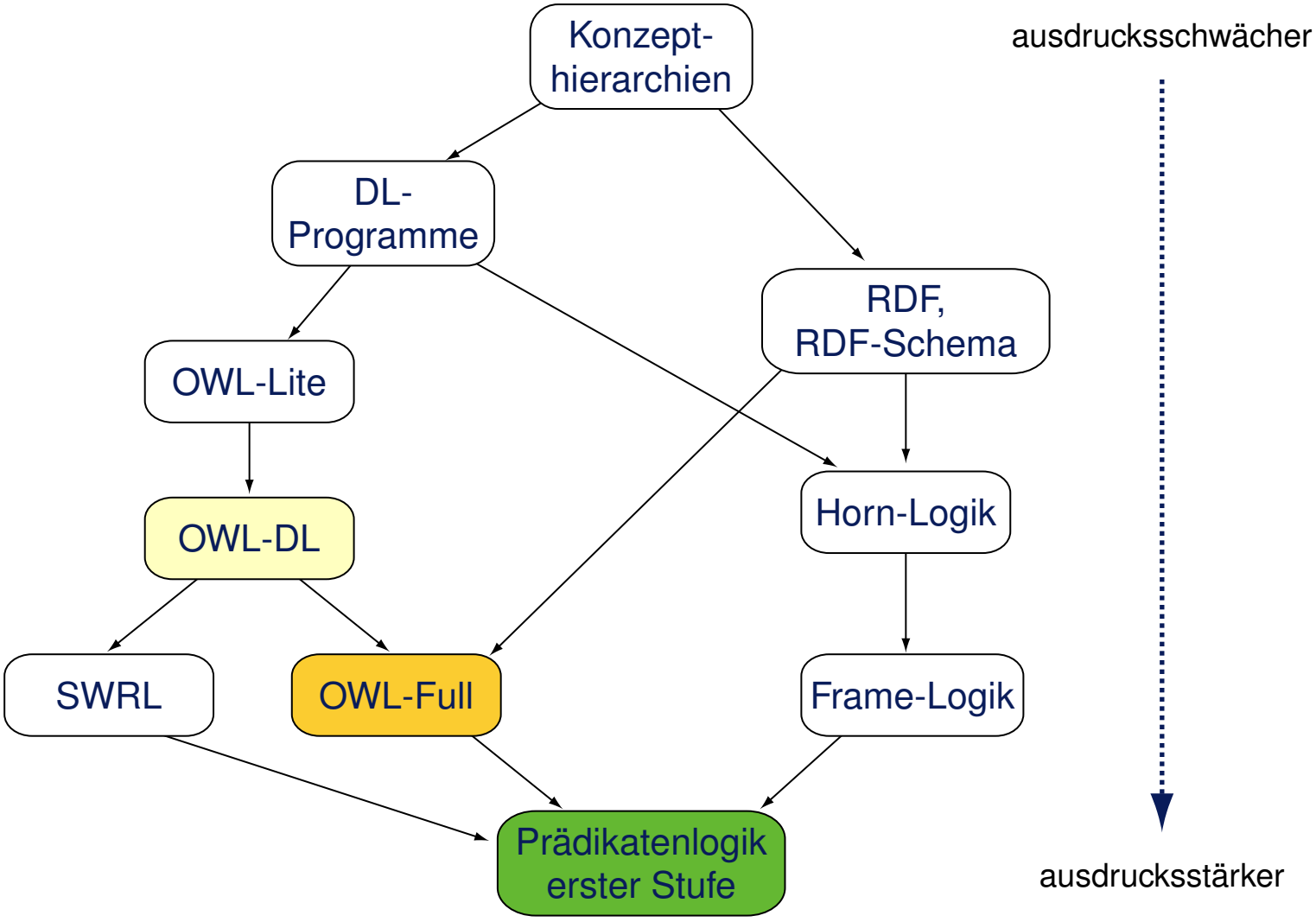
- Die Ausdruckstärke der Prädikatenlogik kann die Modellierung unhandlich machen und Anwender überfordern.
- Die Mächtigkeit und die Freiheiten der Prädikatenlogik erschweren den Abgleich zwischen und den Austausch von Modellen.
- Die Prädikatenlogik ist beweistheoretisch komplex.

➜ Suche nach einer geeigneten Teilmenge der Prädikatenlogik.



OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten (Fortsetzung)



OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten (Fortsetzung)

$$\text{OWL-Lite} \subset \text{OWL-DL} \subset \text{OWL-Full} \subset \text{Prädikatenlogik}$$

OWL-DL:

- keine Reification
- ist entscheidbar
- entspricht der Beschreibungslogik SHOIN(D)

OWL-Full:

- alle Sprachkonstrukte sind verwendbar, solange gültiges RDF vorliegt
- OWL-Full = OWL-DL $\cup$ RDFS bzw. RDFS $\subseteq$ OWL-Full
- erlaubt Reification
- ist nicht entscheidbar
- gilt als „unschöne“ Teilmenge der Prädikatenlogik

Bemerkungen:

- OWL-Full ist für Anwendungen, die maximale Ausdrucksstärke und die syntaktische Freiheit von RDF/RDFS erfordern.
- OWL-Full schafft die Möglichkeit, innerhalb einer Ontologie die Bedeutung der vordefinierten RDF-, RDFS- und OWL-Vokabulare zu erweitern.
- RDF/RDFS ist im Allgemeinen in OWL-Full, und schon nicht mehr in OWL-DL.
- Es ist nicht zu erwarten, dass eine schlussfolgernde Anwendung jedes Konzept von OWL-Full unterstützen kann.

OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten: OWL-DL

Um in der Komplexitätklasse von Beschreibungslogiken zu bleiben, sind folgende Bedingungen einzuhalten [vgl. Antoniou/Harmelen 2004]:

1. Partionierung des Vokabulars.
   Eine Ressource ist entweder eine Klasse oder ein Individuum oder ein Datenwert etc.
2. Explizite Typangaben.

```
</owl:Class rdf:ID="C2"/>

<owl:Class rdf:ID="C1">
  <rdfs:subClassOf rdf:resource="#C2"/>
</owl:Class>
```

3. Wegen Punkt 1. sind folgende Elemente nicht als Datentypen erlaubt:
   - `owl:inverseOf`
   - `owl:FunctionalProperty`
   - `owl:InverseFunctionalProperty`
   - `owl:SymmetricProperty`
4. Kardinalitätsangaben sind in transitiven Properties nicht erlaubt.
5. Eingeschränkte Verwendung anonymer Klassen.

Bemerkungen:

- zu (2) explizite Typangaben:
  Obwohl aus `<rdfs:subClassOf rdf:resource="#C2"/>` folgerbar ist, dass `C2` eine Klasse ist, muss für `C2` z. B. mit `</owl:Class rdf:ID="C2"/>` der Typ „Klasse“ explizit spezifiziert werden.
- OWL-DL ist eine Sprache mit hoher Ausdrucksmächtigkeit bei gleichzeitiger Entscheidbarkeit.
- OWL-DL wurde u. a. auch deswegen spezifiziert, um das existierende Geschäftsfeld und die Befürworter der Beschreibungslogik zu bedienen.

OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten: OWL-Lite

Folgende Sprachkonstrukte sind nur eingeschränkt verwendbar:

- `owl:intersectionOf`
- `owl:minCardinality`
- `owl:maxCardinality`
- `owl:cardinality`

Folgende Sprachkonstrukte können nicht verwendet werden:

- `owl:unionOf`
- `owl:complementOf`
- `owl:oneOf`
- `owl:hasValue`
- `owl:disjointWith`

Bemerkungen:

- OWL-Lite ist für Anwendungen, in denen einfache Klassenhierarchien, Taxonomien und leicht axiomatisierbare Ontologien modelliert werden sollen.
- OWL-Lite stellt eine einfach zu implementierende Untermenge von OWL bereit.
- Einschränkung von Sprachkonstrukten: Argumente der Schnittmengenbildung dürfen nur Klassen und nicht-geschachtelte Restriktionen sein; als Kardinalitätsangaben sind nur die Werte 0 und 1 zugelassen.

OWL: Logikhintergrund

Ausdruckstärke von OWL-Varianten (Fortsetzung)

Worst-Case-Komplexitäten hinsichtlich Inferenz, Erfüllbarkeit, Widersprüchlichkeit einer Formel oder Wissensbasis [Hitzler 2005]:

OWL-Variante	Komplexität
OWL-Lite	Exptime
OWL-DL	NExptime
OWL-Full	unentscheidbar

Insbesondere gilt:

1. Jede OWL-Lite-Ontologie ist auch eine OWL-DL-Ontologie.
   Jede OWL-DL-Ontologie ist auch eine OWL-Full-Ontologie.
2. Jede OWL-Lite-Schlussfolgerung ist auch eine OWL-DL-Schlussfolgerung.
   Jede OWL-DL-Schlussfolgerung ist auch eine OWL-Full-Schlussfolgerung.

OWL: Logikhintergrund

Beschreibungslogik *(Description Logics, DL)*

Beschreibungslogik ist ein Sammelbegriff für verschiedene logikbasierte Sprachen, die mit dem Schwerpunkten „Wissensrepräsentation“ und „Konzeptualisierung“ eines Gegenstandsbereichs entwickelt wurden.

OWL: Logikhintergrund

Beschreibungslogik *(Description Logics, DL)*

Beschreibungslogik ist ein Sammelbegriff für verschiedene logikbasierte Sprachen, die mit dem Schwerpunkten „Wissensrepräsentation“ und „Konzeptualisierung“ eines Gegenstandsbereichs entwickelt wurden.

Charakteristika:

- Teilmenge der Prädikatenlogik erster Stufe (PL1), meist entscheidbar
- Ausdrucksstärke orientiert an praktischer Verwendbarkeit
- Ursprung in semantischen Netzen, u. a. KL-ONE
- Definition eines Vokabulars, bestehend aus Konzepten und Rollen.
  - Konzepte bezeichnen Mengen von Individuen; sie entspechen einstelligen Prädikaten.
  - Rollen bezeichnen Beziehungen zwischen je zwei Individuen; sie entsprechen zweistelligen Prädikaten.
- Konstruktoren für komplexe Konzepte und Rollen auf Basis von atomaren Konzepten und Rollen

OWL: Logikhintergrund

Beschreibungslogik

- ALC ist die kleinste DL, die aussagenlogisch abgeschlossen ist. Die Beschreibungslogik hinter OWL-DL heißt SHOIN(D).
- Konstruktoren sind $\wedge$, $\vee$ und $\neg$, geschrieben als $\sqcap$, $\sqcup$, und $\neg$.
- Mittels der Quantoren $\forall$ und $\exists$ werden Rollen eingeschränkt:
  Beispiel: $\forall$hasChild.Female
  = diejenigen x, bei denen alle Kinder weiblich sind = die nur Töchter haben.
- Kardinalitätsangaben für Rollen und Konzepte
- Datentypen:
  Beispiel: hasAge.($\geq$21)
- inverse Rollen:
  Beispiel: hasChild$^-$ $\equiv$ hasParent
- transitive Rollen:
  Beispiel: hasAncestor*(descendant)
- Rollenkomposition:
  Beispiel: hasParent.hasBrother(uncle)

OWL: Logikhintergrund

Beschreibungslogik: Wissensbasis

Die Formeln einer DL-Wissensbasis werden oft in zwei Mengen aufgeteilt:

TBox Axiome, um Prinzipien des Gegenstandsbereichs zu beschreiben:

$\text{Lecturer} \sqsubseteq \text{TeachingStaff}$

ABox Axiome, um Ausprägungen des Gegenstandsbereichs zu beschreiben:

$\text{Lecturer}(\text{Benno-Stein})$

OWL: Logikhintergrund

Beschreibungslogik: Wissensbasis

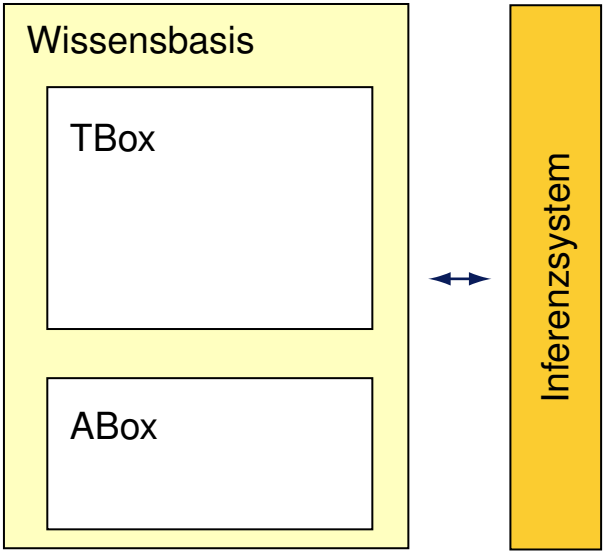
Die Formeln einer DL-Wissensbasis werden oft in zwei Mengen aufgeteilt:

TBox Axiome, um Prinzipien des Gegenstandsbereichs zu beschreiben:

Lecturer $\sqsubseteq$ TeachingStaff

ABox Axiome, um Ausprägungen des Gegenstandsbereichs zu beschreiben:

Lecturer(Benno-Stein)



OWL: Logikhintergrund

Beschreibungslogik: Wissensbasis

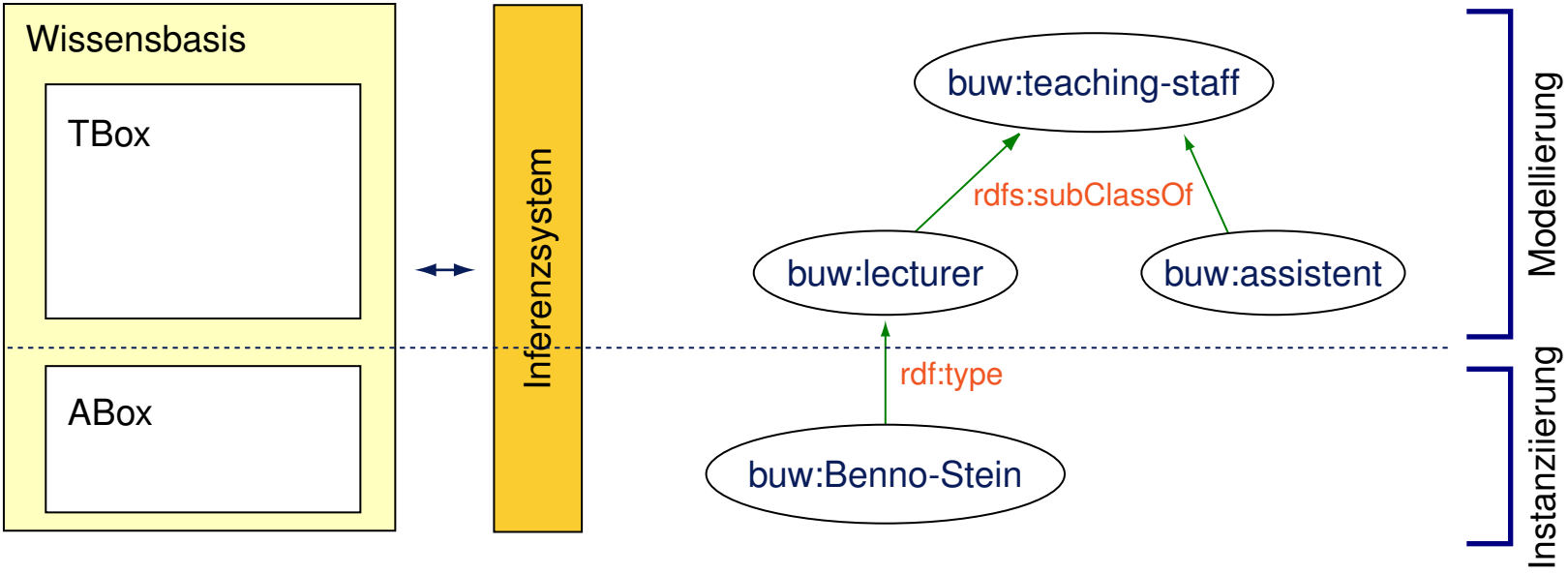
Die Formeln einer DL-Wissensbasis werden oft in zwei Mengen aufgeteilt:

TBox Axiome, um Prinzipien des Gegenstandsbereichs zu beschreiben:

Lecturer $\sqsubseteq$ TeachingStaff

ABox Axiome, um Ausprägungen des Gegenstandsbereichs zu beschreiben:

Lecturer(Benno-Stein)



Bemerkungen:

- Die TBox beinhaltet die terminologische Beschreibung *(terminology)* – also die Axiomatisierung bzw. das Modell des Gegenstandsbereichs; die ABox beinhaltet konkrete Behauptungen *(assertions)*.
- Aus prädikatenlogischer Sicht entspricht die Unterscheidung einer TBox und einer ABox der Aufteilung einer Formelmenge in Formeln mit und ohne Variablen. Sie ist aus logischen Gründen nicht notwendig.

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$	Lecturer $\sqcup$ Assistent

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$	Lecturer $\sqcup$ Assistent
`complementOf`	$\neg C$	$\neg C(x)$	$\neg$ Female
`oneOf`	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	$x = x_1 \vee \ldots \vee x = x_n$	{Alice} $\sqcup$ {Bob}

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$	Lecturer $\sqcup$ Assistent
`complementOf`	$\neg C$	$\neg C(x)$	$\neg$ Female
`oneOf`	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	$x = x_1 \vee \ldots \vee x = x_n$	{Alice} $\sqcup$ {Bob}
`allValuesFrom`	$\forall P.C$	$\forall y : P(x, y) \rightarrow C(y)$	$\forall$hasChild.Male
`someValuesFrom`	$\exists P.C$	$\exists y : P(x, y) \wedge C(y)$	$\exists$hasChild.Male

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$	Lecturer $\sqcup$ Assistent
`complementOf`	$\neg C$	$\neg C(x)$	$\neg$ Female
`oneOf`	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	$x = x_1 \vee \ldots \vee x = x_n$	{Alice} $\sqcup$ {Bob}
`allValuesFrom`	$\forall P.C$	$\forall y : P(x,y) \rightarrow C(y)$	$\forall$hasChild.Male
`someValuesFrom`	$\exists P.C$	$\exists y : P(x,y) \wedge C(y)$	$\exists$hasChild.Male
`maxCardinality`	$\leq nP$	$\exists^{\leq n} y : P(x,y)$	$\leq n$hasChild
`minCardinality`	$\geq nP$	$\exists^{\geq n} y : P(x,y)$	$\geq n$hasChild

OWL: Logikhintergrund

Beschreibungslogik: wichtige Sprachelemente

OWL-Name	DL-Syntax	PL-Syntax	Beispiel in DL-Syntax
`intersectionOF`	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$	Lecturer $\sqcup$ Assistent
`complementOf`	$\neg C$	$\neg C(x)$	$\neg$ Female
`oneOf`	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	$x = x_1 \vee \ldots \vee x = x_n$	{Alice} $\sqcup$ {Bob}
`allValuesFrom`	$\forall P.C$	$\forall y : P(x,y) \rightarrow C(y)$	$\forall$hasChild.Male
`someValuesFrom`	$\exists P.C$	$\exists y : P(x,y) \wedge C(y)$	$\exists$hasChild.Male
`maxCardinality`	$\leq nP$	$\exists^{\leq n} y : P(x,y)$	$\leq n$hasChild
`minCardinality`	$\geq nP$	$\exists^{\geq n} y : P(x,y)$	$\geq n$hasChild
`subClassOf`	$C_1 \sqsubseteq C_2$	$\forall x : C_1(x) \rightarrow C_2(x)$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`equivalentClass`	$C_1 \equiv C_2$	$\forall x : C_1(x) \leftrightarrow C_2(x)$	Man $\equiv$ Human $\sqcap$ Male

Bemerkungen:

- $\forall P.C$ bzw. $\forall y : P(x, y) \rightarrow C(y)$ bezeichnet folgende Menge:

$$\{x \mid \text{ wenn } (x, y) \in P \text{ dann } y \in C\}$$

- $\exists P.C$ bzw. $\exists y : P(x, y) \wedge C(y)$ bezeichnet folgende Menge:

$$\{x \mid (x, y) \in P \text{ und } y \in C\}$$

OWL: Logikhintergrund

Beschreibungslogik: Inferenzprobleme [vgl. Hitzler 2005]

1. Wissensbasiskonsistenz. Ist Wissensbasis KB widerspruchsfrei?
   $KB \models$ false ?

2. Klassenkonsistenz. Ist Klasse C leer?
   $C \equiv \bot$?

3. Subsumption. Ist C Unterklasse von D?
   $C \sqsubseteq D$?

4. Klassenäquivalenz. Sind die Klassen C und D identisch?
   $C \equiv D$?

5. Klassendisjunktheit. Sind die Klassen C und D disjunkt?
   $C \sqcup D = \bot$?

6. Klassenzugehörigkeit. Ist Individuum a aus Klasse C?
   $C(a)$?

OWL: Logikhintergrund

Gegenüberstellung von Sprachen zur Wissensformulierung

Formulierung in natürlicher Sprache:

„Dozenten bilden einen Teil des Lehrpersonals.“

Formulierung in Prädikatenlogik:

$$\forall x : \textit{Lecturer}(x) \rightarrow \textit{TeachingStaff}(x)$$

Formulierung in Beschreibungslogik:

$$\text{Lecturer} \sqsubseteq \text{TeachingStaff}$$

Formulierung in RDF/XML:

```
<owl:Class rdf:about="http://www.buw.de/lecturer">
  <rdfs:subClassOf rdf:resource="http://www.buw.de/teaching-staff"/>
</owl:Class>
```

OWL

Quellen zum Nachlernen und Nachschlagen im Web

- W3C OWL-Übersichtseite. www.w3.org/2004/OWL
- D.L. McGuinness, F. van Harmelen, Eds. *OWL Web Ontology Language Overview.* W3C Recommendation. www.w3.org/TR/owl-features
- Smith, Welty, McGuinness, Eds. *OWL Web Ontology Language Guide.* W3C Recommendation. www.w3.org/TR/owl-guide, www.semaweb.org/dokumente/w3/TR/2004/REC-owl-guide-20040210-DE.html
- Dean, Schreiber, Eds. *OWL Web Ontology Language Reference.* W3C Recommendation. www.w3.org/TR/owl-ref
- Patel-Schneider, Hayes, Horrocks, Eds. *OWL Web Ontology Language Semantics and Abstract Syntax.* W3C Recommendation. www.w3.org/TR/owl-semantics
- DAML Homepage. www.daml.org
- OIL Homepage. www.ontoknowledge.org/oil
- HP Labs Semantic Web Research. www.hpl.hp.com/semweb
- WonderWeb Homepage. wonderweb.semanticweb.org

OWL

Quellen zum Nachlernen und Nachschlagen im Web (Fortsetzung)

Anwendung und Implementierung:

- Jena. Semantic Web Framework for Java. jena.sourceforge.net
- Protégé Ontology Editor. protege.stanford.edu
- BBN Ontology Validator. owl.bbn.com/validator
- WonderWeb Ontology Validator. phoebus.cs.man.ac.uk:9999/OWL/Validator
- Knublauch, Oberle, Tetlow, Wallace, Eds.
  A Semantic Web Primer for Object-Oriented Software Developers.
  Suggested W3C Note. www.w3.org/2001/sw/BestPractices/SE/ODSD

Beschreibungslogik:

- F. Baader, W. Nutt. Basic Description Logics.
  www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-02.pdf
- E. Franconi. Tutorials für Aussagen-, Prädikaten- und Beschreibungslogik.
  www.inf.unibz.it/~franconi/dl/course