Grundlagen der Künstlichen Intelligenz - Informatik

Rapp, DHBW Lörrach

20.10.2023

Inhaltsübersicht

Lernziele

Meine 3 Lernziele für heute

1. Ich kenne die grundlegenden Begriffe des Sematic Web und kann diese in einen Kontext setzen.
2. Ich bin vertraut im Umgang mit RDF und RDF Schema in einfachen Ontologien.
3. Durch die Anwendung grundlegender SPARQL-Syntax kann ich in OWL-Ontologien enthaltenes Wissen abfragen.

Einführung in das Semantische Web

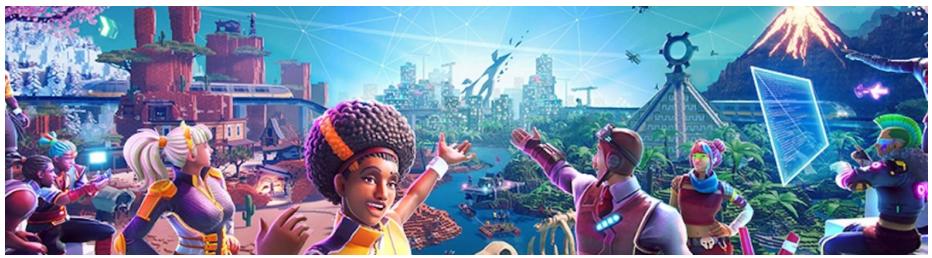
Das Web

Das Web flankiert den Übergang von der Industrie- zur Informationsgesellschaft und bietet die Infrastruktur für eine neue Qualität des Umgangs mit Informationen hinsichtlich Beschaffung wie auch Bereitstellung.

- hohe Verfügbarkeit
- hohe Aktualität
- geringe Kosten

Weitere Lebensbereiche werden “webisiert“:

- Behörden, Verwaltung (eGovernment)
- Ausbildung (eLearning, eEducation)
- Sozialkontakte (Social-Networking-Plattformen, Partnerbörsen)
- Alltag?



“Metaverse“

Wiederholung Syntax vs. Semantik

Syntax

- Zusammenstellung, Satzbau (griech.)
- steht für die Struktur von Daten, d.h. charakterisiert, was "wohlgeformte" Daten sind.

Semantik

- zum Zeichen gehörend (griech.)
- steht für die Bedeutung von Daten, d.h. charakterisiert beispielsweise, welche inhaltlichen Schlussfolgerungen sich ziehen lassen.

Beispiele

$4+) = ($	$3 + 4 = 12$	$3 + 4 = 7$
syntaktisch falsch	syntaktisch richtig	syntaktisch richtig
-	semantisch falsch	semantisch richtig

Lösungsansätze im Semantic Web

Probleme des Web

- Lokalisierung von Informationen problematisch
- Heutige Suchmaschinen gut, aber teilweise stichwortbasiert
- wünschenswert: inhaltliche und semantische Suche

Lösungsansätze

1. Ad hoc: Verwendung von KI-Methoden zur Auswertung bestehender unstrukturierter Informationen im Web
2. A priori: Strukturierung der Web-Informationen zur Erleichterung der automatisierten Auswertung

Literaturempfehlungen

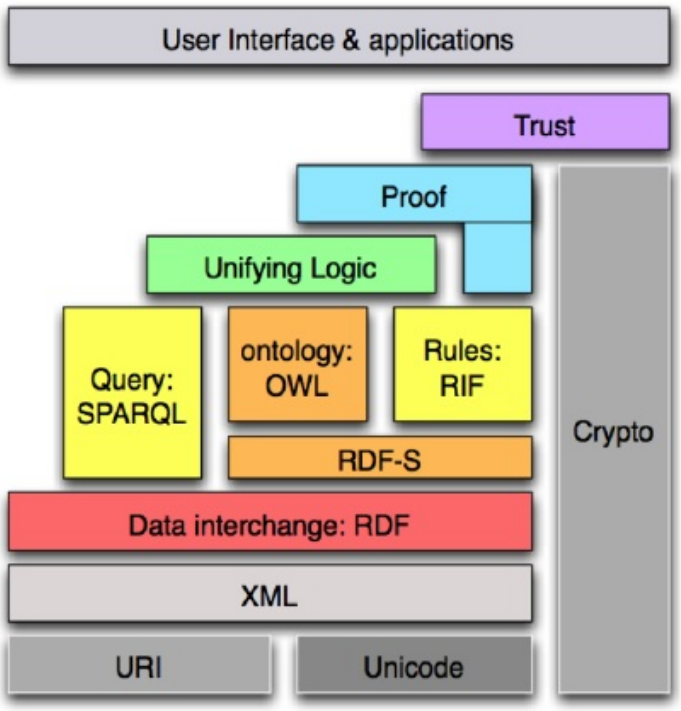
Empfehlung

- Semantic Web Grundlagen, Hitzler et al. (2008), Springer-Verlag

Heutige Folien angelehnt an

- Foundations of Semantic Web Technologies, Hitzler et al. (2009), CRC Press
- Semantic Web Grundlagen, Folien von Prof. Dr. Birte Glimm, Uni Ulm

Semantic Web - Überblick der Standards



W3C Semantic Web

1994	First public presentation of the Semantic Web idea
1998	Start of standardization of data model (RDF) and a first ontology languages (RDFS) at W3C
2000	Start of large research projects about ontologies in the US and Europe (DAML & Ontoknowledge)
2002	Start of standardization of a new ontology language (OWL) based on research results
2004	Finalization of the standard for data (RDF) and ontology (OWL)
2008	Standardization of a query language (SPARQL)
2009	Extension of OWL to OWL 2.0
2010	Standard Rule Interchange Format (RIF)

URIs

URIs - Idee

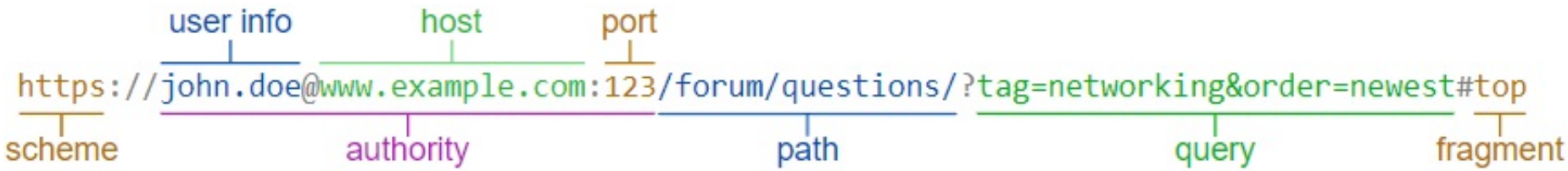
Uniform Resource Identifier (URI)

- dienen der weltweit eindeutigen Identifizierung von abstrakten oder physischen Ressourcen
- Ressource kann jedes Objekt sein, das (im Kontext der gegebenen Anwendung) eine klare Identität besitzt (z.B. Bücher, Orte, Menschen, Verlage, aber auch Beziehungen zwischen diesen Dingen)
- In bestimmten Domänen bereits Ähnliches realisiert: ISBN für Bücher

URIs

Syntax

- Erweiterung des URL-Konzeptes
- nicht jede URI bezeichnet aber ein Webdokument (umgekehrt wird als URI für Webdokumente häufig deren URL verwendet)
- Beginnt mit dem sogenannten URI-Schema, das durch einen Doppelpunkt (:) vom nachfolgenden Teil getrennt ist (z.B. https, ftp, mailto)
- Häufig hierarchisch aufgebaut

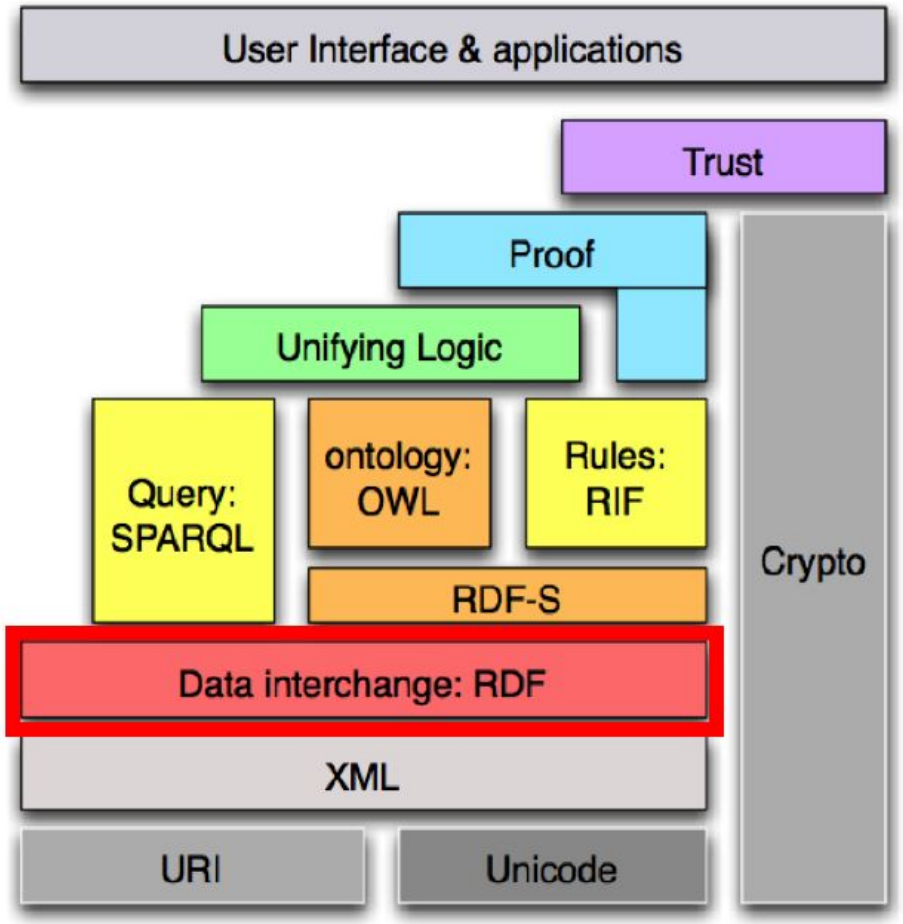


Selbst definierte URIs

- Nötig, wenn für eine Ressource (noch) keine URI existiert (bzw. bekannt ist)
- Strategie zur Vermeidung von (ungewollten) Überschneidungen: Nutzung von http-URIs einer eigenen Webseite

RDF Datenmodell

Einführung in RDF



Unzulänglichkeiten von XML

Unzulänglichkeiten von XML

- Tag-Namen mehrdeutig (durch Namespaces und URIs behebbar)
- Baumstruktur nicht optimal für
  - intuitive Beschreibung der Daten
  - Informationsintegration

Beispiel: Wie kodiert man in einem Baum den Fakt

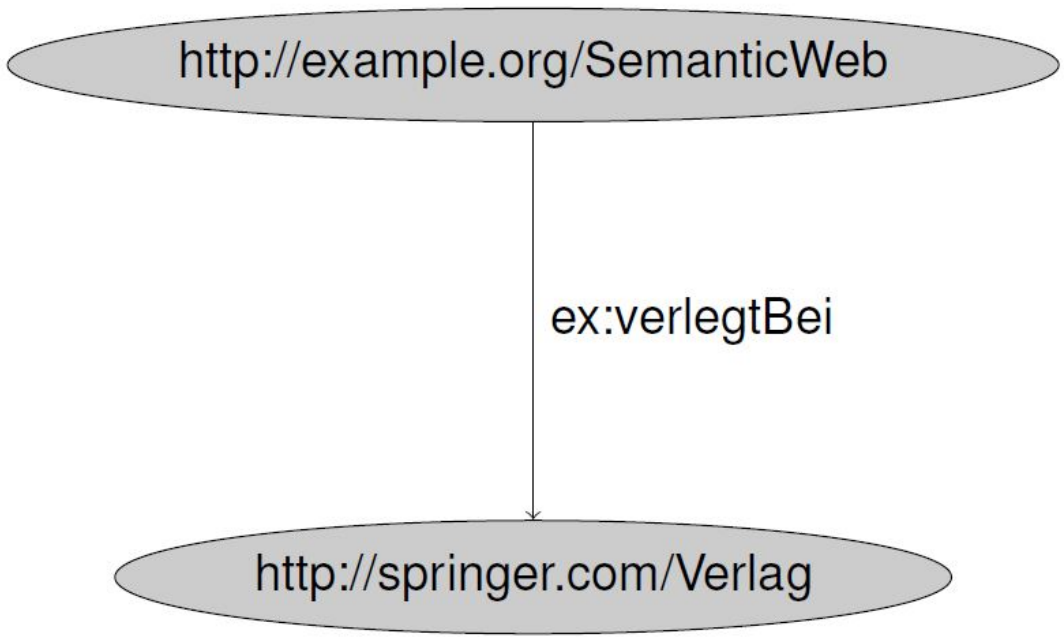
"Das Buch 'Semantic Web - Grundlagen' wird beim Springer-Verlag verlegt."

```
<Verlegt>
  <Verlag>Springer-Verlag</Verlag>
  <Buch>Semantic Web -- Grundlagen</Buch>
</Verlegt>

<Verlag Name="Springer-Verlag">
  <Verlegt Buch="Semantic Web -- Grundlagen"/>
</Verlag>

<Buch Name="Semantic Web -- Grundlagen">
  <Verleger Verlag="Springer-Verlag"/>
</Buch>
```

RDF: Graphen statt Bäume

Lösungsansatz

Darstellung durch gerichtete Graphen:

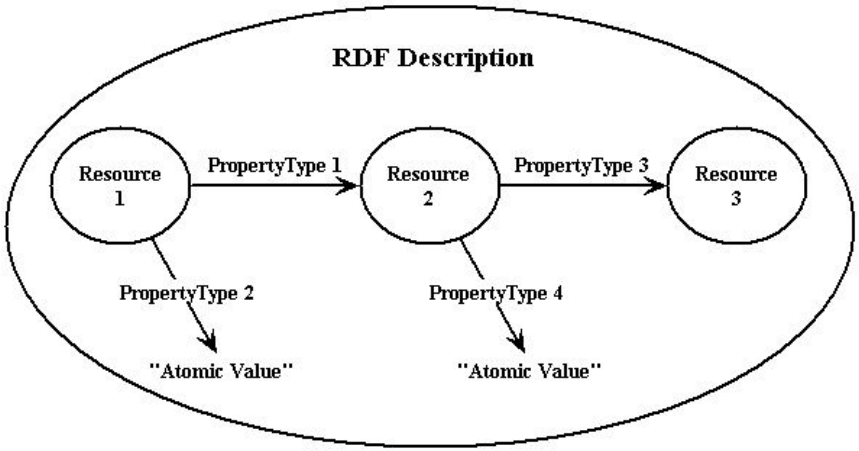


Allgemeines zu RDF

Resource Description Framework (RDF)

- RDF ist ein **Datenmodell für Ressourcen**
  - kodiert strukturierte Informationen
  - universelles, maschinenlesbares Austauschformat
- W3C Empfehlung (http://www.w3.org/RDF)
- Grundlegender Baustein des Semantischen Webs

Ressourcen in RDF werden eindeutig durch URIs beschrieben:

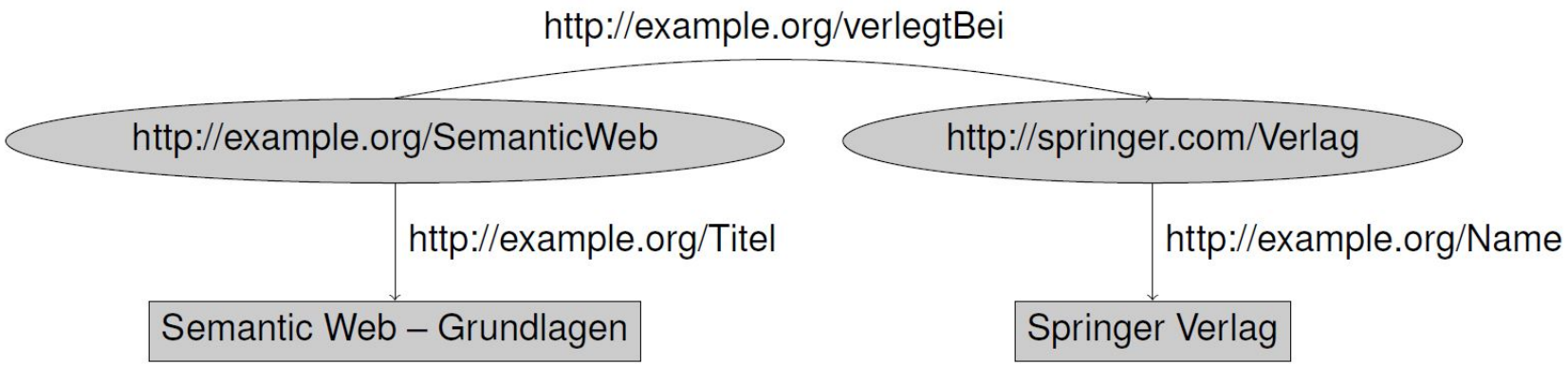


Ressourcen haben **Eigenschaften** und Eigenschaften werden durch ihre **Typen** charakterisiert, die wiederum verschiedene **Werte (Literale)** annehmen können.

Literale

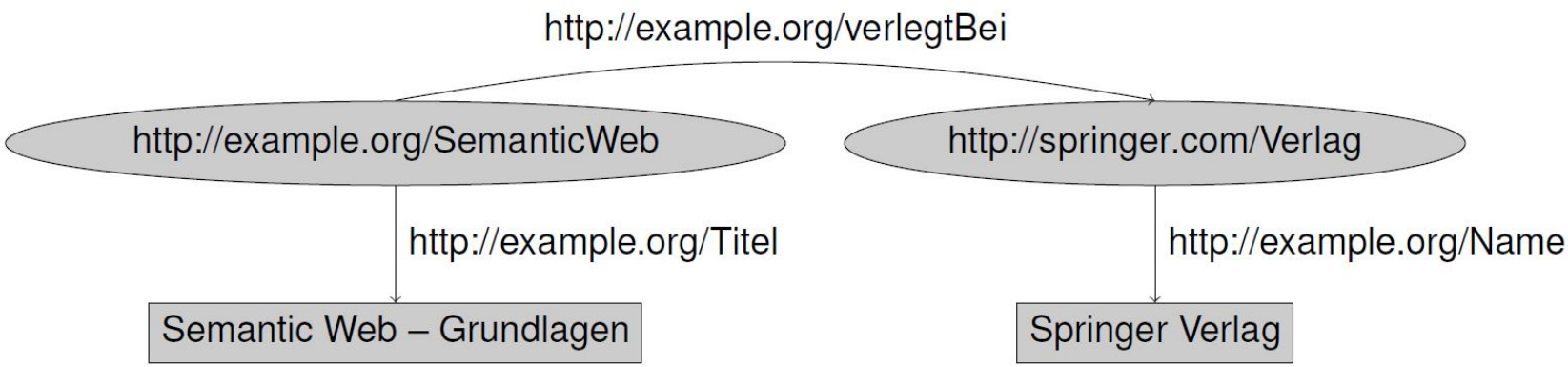
Literale

- Zur Repräsentation von Datenwerten
- Darstellung als Zeichenketten
- Interpretation erfolgt durch **Datentyp**
- Literale ohne Datentyp werden wie Zeichenketten behandelt



RDF Graph

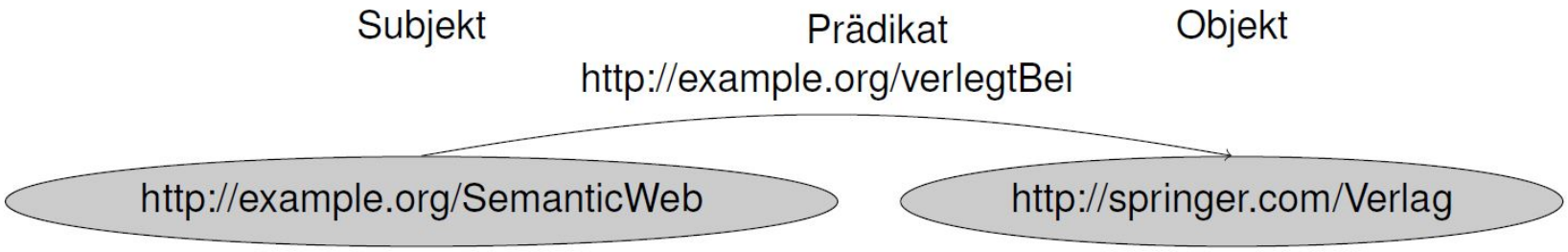
Ein **RDF Graph** ist eine Menge von **RDF Tripeln.**
Für Graphen existieren verschiedene Darstellungsmöglichkeiten, z.B.



Visualisierung als Knoten-Kante-Knoten-Tripel

RDF-Tripel

Bestandteile eines RDF-Tripels:



Angelehnt an linguistische Kategorien, aber nicht immer stimmig.

Erlaubte Belegungen

- **Subjekt:** URI oder leerer Knoten (d.h. keine URI oder Literal angegeben)
- **Prädikat:** URI (auch Property genannt)
- **Objekt:** URI oder leerer Knoten oder Literal

Knoten- und Kantenbezeichner sind eindeutig, daher ist der ursprüngliche Graph aus einer Tripel-Liste rekonstruierbar.

RDF-Syntax Turtle

Turtle als einfache Syntax für RDF

Turtle

Serialisierung für Graphen im Resource Description Framework, die für Menschen einfach lesbar ist.

Direkte Auflistung der Tripel

- N3: “Notation 3“ - umfangreicher Formalismus
- N-Triples: Teil von N3
- Turtle: Kann als Erweiterung von N-Triples aufgefasst werden

Turtle Syntax mit Beispiel (1/2)

Syntax in Turtle

- URIs in spitzen Klammern
- Literale in Anführungszeichen
- Tripel durch Punkt abgeschlossen
- Leerzeichen und Zeilenumbrüche außerhalb von Bezeichnern werden ignoriert

Beispiel

```
<http://ex.org/SemanticWeb> <http://ex.org/verlegtBei> <http://springer.com/Verlag> .
<http://ex.org/SemanticWeb> <http://ex.org/Titel> "Semantic Web Grundlagen" .
<http://springer.com/Verlag> <http://ex.org/Name> "Springer Verlag" .
```

Abkürzungen für Präfixe im obigen Beispiel

```
@prefix ex: <http://ex.org/> .
@prefix springer: <http://springer.com/> .
ex:SemanticWeb ex:verlegtBei springer:Verlag .
ex:SemanticWeb ex:Titel "Semantic Web Grundlagen" .
springer:Verlag ex:Name "Springer Verlag" .
```

Turtle Syntax mit Beispiel (2/2)

Mehrere Tripel mit gleichem Subjekt kann man zusammenfassen:

```
@prefix ex: <http://ex.org/> .
@prefix springer: <http://springer.com/> .
ex:SemanticWeb     ex:verlegtBei springer:Verlag ;
                   ex:Titel "Semantic Web Grundlagen" .
springer:Verlag ex:Name "Springer Verlag" .
```

Ebenso Tripel mit gleichem Subjekt und Prädikat:

```
@prefix ex: <http://ex.org/> .
ex:SemanticWeb ex:Autor  ex:Hitzler, ex:Kroetzsch, ex:Rudolph, ex:Sure ;
               ex:Titel "Semantic Web Grundlagen" .
```

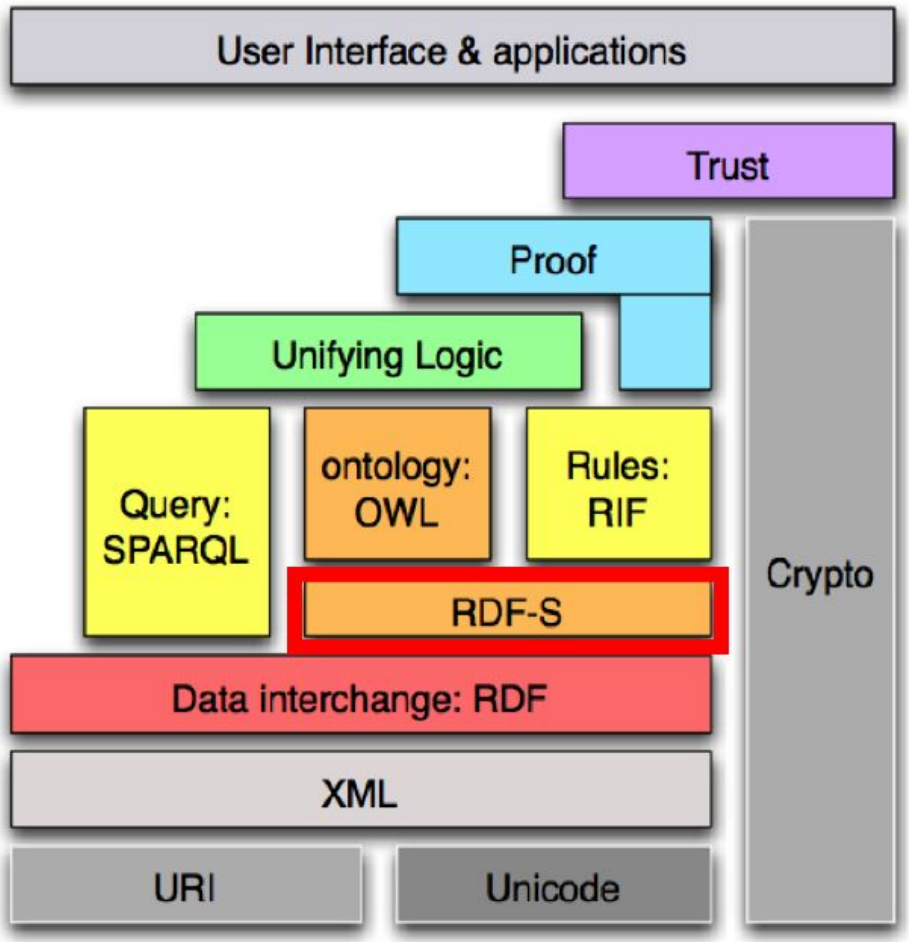
Einsatz von Turtle vs. XML

- Turtle intuitiv gut lesbar und maschinenverarbeitbar
- Aber: bessere Tool-Unterstützung und Programmbibliotheken für XML

⇒ **XML-Syntax für RDF am verbreitetsten**

RDF Schema

RDF-S


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

Bewertung und Unzulänglichkeiten von RDF

Bewertung von RDF

- Weitläufig unterstützter Standard für Speicherung und Austausch von Daten
- Ermöglicht weitgehend syntaxunabhängige Darstellung verteilter Informationen in graphbasiertem Datenmodell

Reines RDF

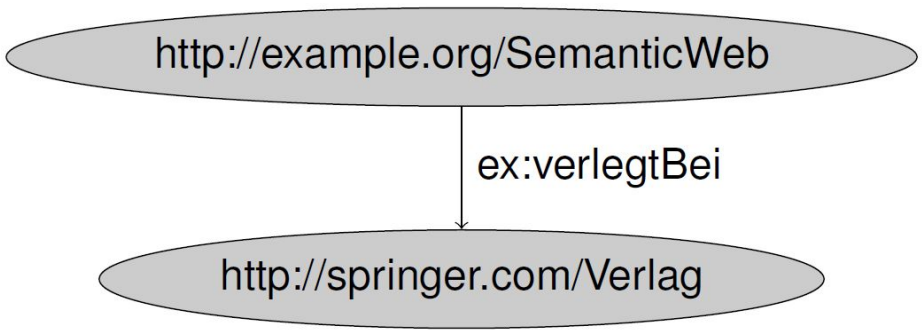
- Sehr “individuenorientiert“
- Kaum Möglichkeiten zur Kodierung von Schemawissen

$\Rightarrow$ **RDF Schema**

Schemawissen mit RDFS

RDF

RDF bietet universelle Möglichkeit zur **Kodierung faktischer Daten** im Web:



D.h. es können Aussagen über einzelne Ressourcen (Individuen) und deren Beziehungen gemacht werden.

Wünschenswert

Aussagen über generische Mengen von Individuen (=Klassen), z.B. Verlage, Organisationen, Personen etc.

Schemawissen mit RDFS

Weiterhin wünschenswert

Spezifikation der logischen Zusammenhänge zwischen Individuen, Klassen und Beziehungen, um möglichst viel Semantik des Gegenstandsbereiches einzufangen, z.B.

- "Verlage sind Organisationen."
- "Nur Personen schreiben Bücher."

In Datenbanksprache ausgedrückt: **Schemawissen**

RDFS Überblick

RDF Schema (RDFS)

- Teil der W3C Empfehlung zu RDF
- Ermöglicht Spezifikation von schematischem (auch: terminologischen) Wissen
- Spezielles RDF-Vokabular (also: jedes RDFS-Dokument ist ein RDF-Dokument)

Namensraum

`http://www.w3.org/2000/01/rdf-schema#`
i.d.R. abgekürzt mit *rdfs*

Auszug:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
        dc:title "The RDF Schema vocabulary (RDFS)" .

rdfs:Resource a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Resource" ;
        rdfs:comment "The class resource, everything." .

rdfs:Class a rdfs:Class ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "Class" ;
        rdfs:comment "The class of classes." ;
        rdfs:subClassOf rdfs:Resource .

rdfs:subClassOf a rdf:Property ;
        rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
        rdfs:label "subClassOf" ;
        rdfs:comment "The subject is a subclass of a class." ;
        rdfs:range rdfs:Class ;
        rdfs:domain rdfs:Class .
```

RDFS Ontologiesprache

RDF Schema

- Vokabular nicht themengebunden (wie z.B. bei FOAF), sondern generisch
- Vorteil: jede Software mit RDFS-Unterstützung interpretiert jedes mittels RDFS definierte Vokabular korrekt
- Funktionalität macht RDFS zu einer **Ontologiesprache** für leichtgewichtige *(engl. lightweight)* Ontologien

"A little semantics goes a long way"

Zitat des KI-Forschers James A. Hendler

Klassen

Klassen und Instanzen am Beispiel

RDF-Tripel Beispiel

```
ex:SemanticWeb rdf:type ex:Lehrbuch .
```

Dem **Subjekt** *"Semantic Web - Grundlagen"* wird durch das **Prädikat** `rdf:type` (Abk. *a*) das **Objekt** *Lehrbuch* als *Typ* zugewiesen.

Interpretation im RDF Schema

- Das Objekt "Lehrbuch" wird als **Klasse** interpretiert.
- Das Subjekt *"Semantic Web - Grundlagen"* wird als **Instanz der Klasse** "Lehrbuch" interpretiert.

Zugehörigkeit zu weiteren Klassen
Klassenzugehörigkeit ist nicht exklusiv, z.B. gleichzeitig möglich:

```
ex:SemanticWeb rdf:type ex:Unterhaltsam .
```

Unterklassen in RDFS

Unterklassen werden realisiert durch die Property:

`rdfs:subClassOf`

Beispiel

```
ex:Lehrbuch rdfs:subClassOf ex:Buch .
```

"Die Klasse der Lehrbücher ist eine Unterklasse der Klasse der Bücher."

Intuitive Parallele zur Mengenlehre

- `rdf:type` entspricht $\in$
- `rdf:subClassOf` entspricht $\subseteq$

Klassenhierarchien *(auch: Taxonomien)*

Üblich: nicht nur einzelne Unterklassenbeziehungen, sondern ganze **Klassenhierarchien.**

Beispiel

```
ex:Lehrbuch rdfs:subClassOf ex:Buch .
ex:Buch rdfs:subClassOf ex:Printmedium .
ex:Zeitschrift rdfs:subClassOf ex:Printmedium .
```

In RDFS-Semantik verankert: Transitivität der `rdfs:subClassOf`-Property, d.h. es folgt automatisch:

```
ex:Lehrbuch rdfs:subClassOf ex:Printmedium .
```

Properties

Property

- auch: Relation, Beziehung
- Property-Bezeichner in Tripeln üblicherweise an Prädikatsstelle
- Charakterisieren, auf welche Art zwei Ressourcen zueinander in Beziehung stehen
- Mathematisch oft dargestellt als Menge von Paaren: $verheiratetMit = \{(Brad, Angelina), ...\}$
- URI wird als Property-Bezeichner gekennzeichnet durch entsprechende Typung:

```
ex:verlegtBei rdf:type rdf:Property .
```

Einfache Ontologien

RDFS für einfache Ontologien

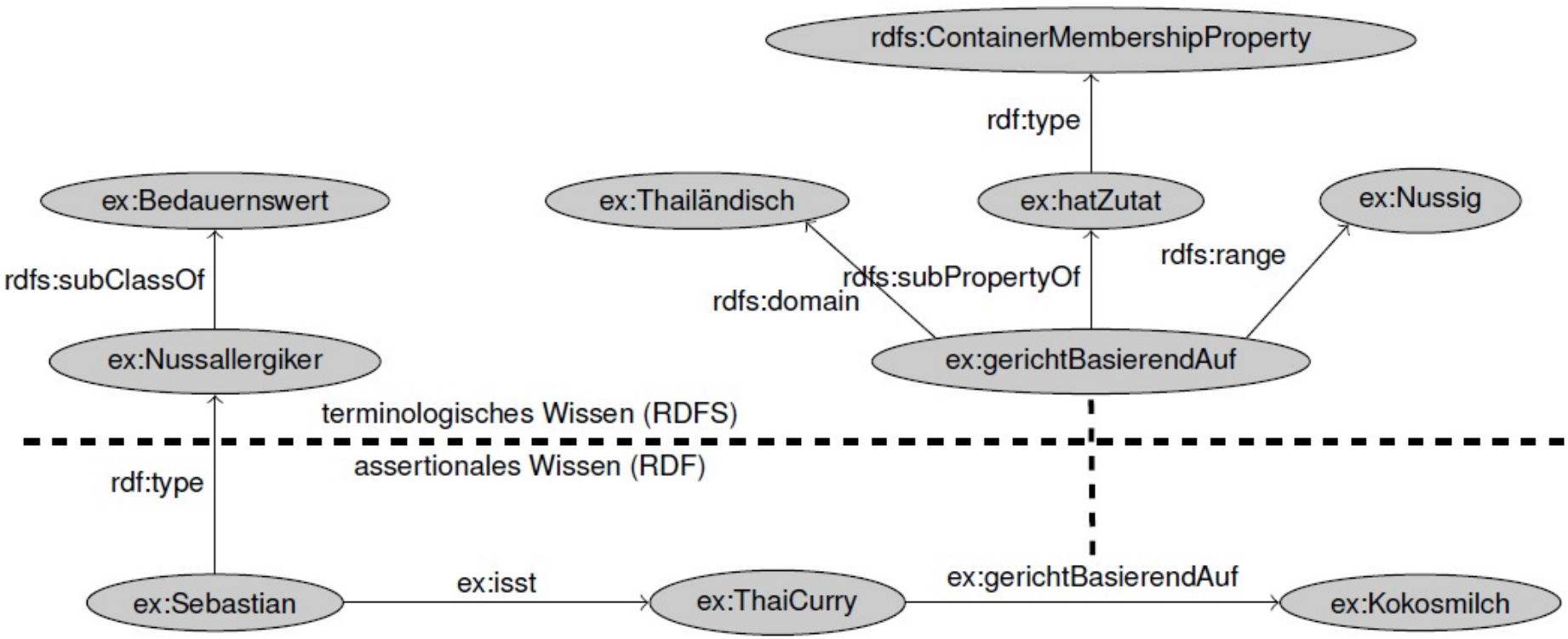
Einfache Ontologien

- Mit den durch RDFS bereitgestellten Sprachmitteln können bestimmte Gegenstandsbereiche bereits in wichtigen Aspekten semantisch erfasst werden
  Und:
- Auf der Basis der speziellen Semantik von RDFS kann schon ein gewisses Maß impliziten Wissens geschlussfolgert werden

RDFS stellt eine, wenn auch noch vergleichsweise wenig ausdrucksstarke, **Ontologiesprache** dar.

Thai-Curry als Beispiel einer einfachen Ontologie

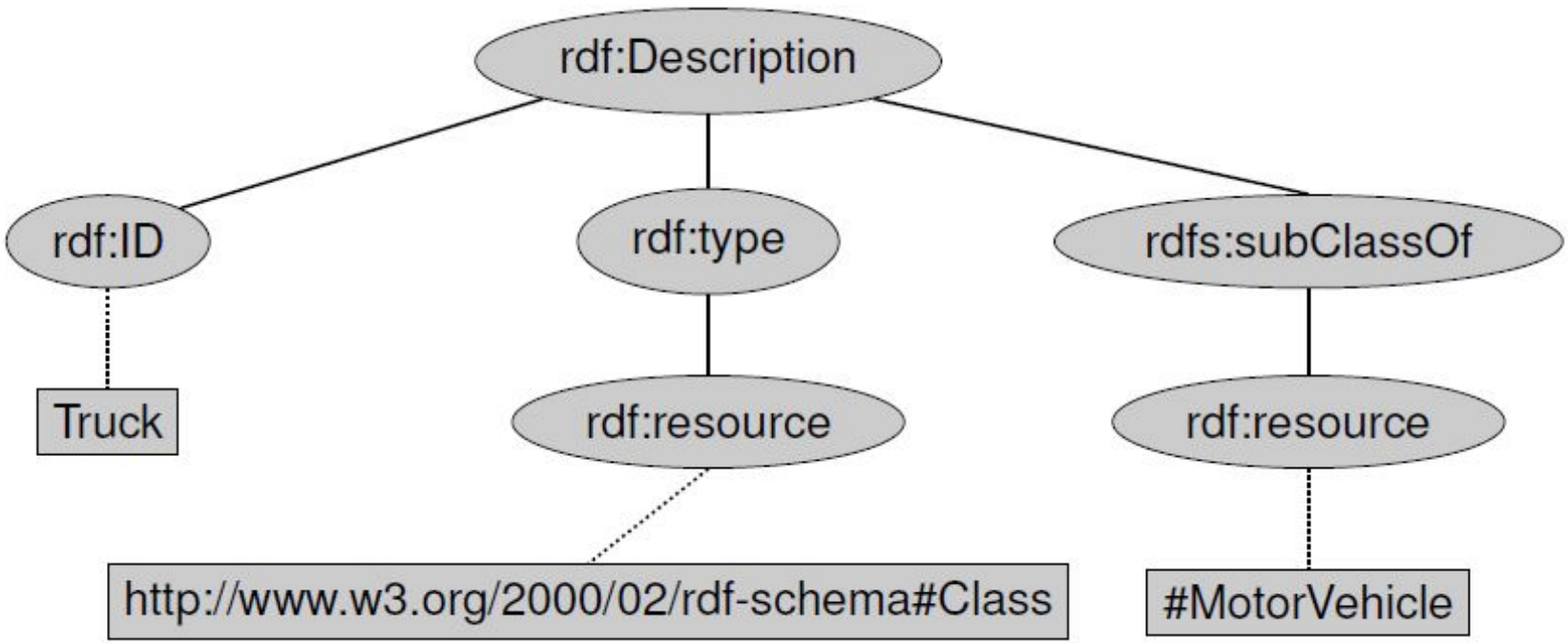
```
ex:ThaiCurry              ex:gerichtBasierendAuf ex:Kokosmilch .
ex:Sebastian              rdf:type               ex:Nussallergiker .
ex:Sebastian              ex:isst                ex:ThaiCurry .
ex:Nussallergiker         rdfs:subClassOf        ex:Bedauernswert .
ex:gerichtBasierendAuf    rdfs:domain            ex:Thailändisch .
ex:gerichtBasierendAuf    rdfs:range             ex:Nussig .
ex:gerichtBasierendAuf    rdfs:subPropertyOf     ex:hatZutat .
ex:hatZutat               rdf:type               rdfs:ContainerMembershipProperty .
```



1 XML Dokument - 3 Interpretationen (1/3 - XML)

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Interpretation als XML



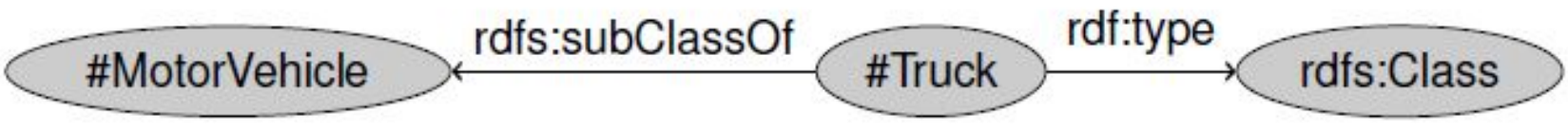
1 XML Dokument - 3 Interpretationen (2/3 - RDF)

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

Interpretation als RDF

- Anderes Datenmodell
- `rdf:Description`, `rdf:ID` und `rdf:resource` haben eine festgelegte Bedeutung

subject	predicate	object
#Truck	rdf:type	rdfs:Class
#Truck	rdfs:subClassOf	#Motorvehicle

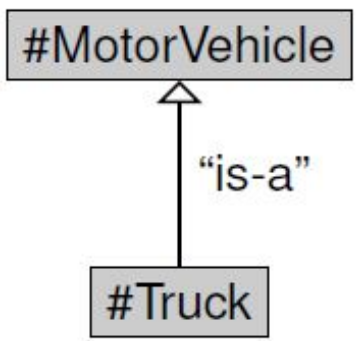


1 XML Dokument - 3 Interpretationen (3/3 - RDF Schema)

```
<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource=
    "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

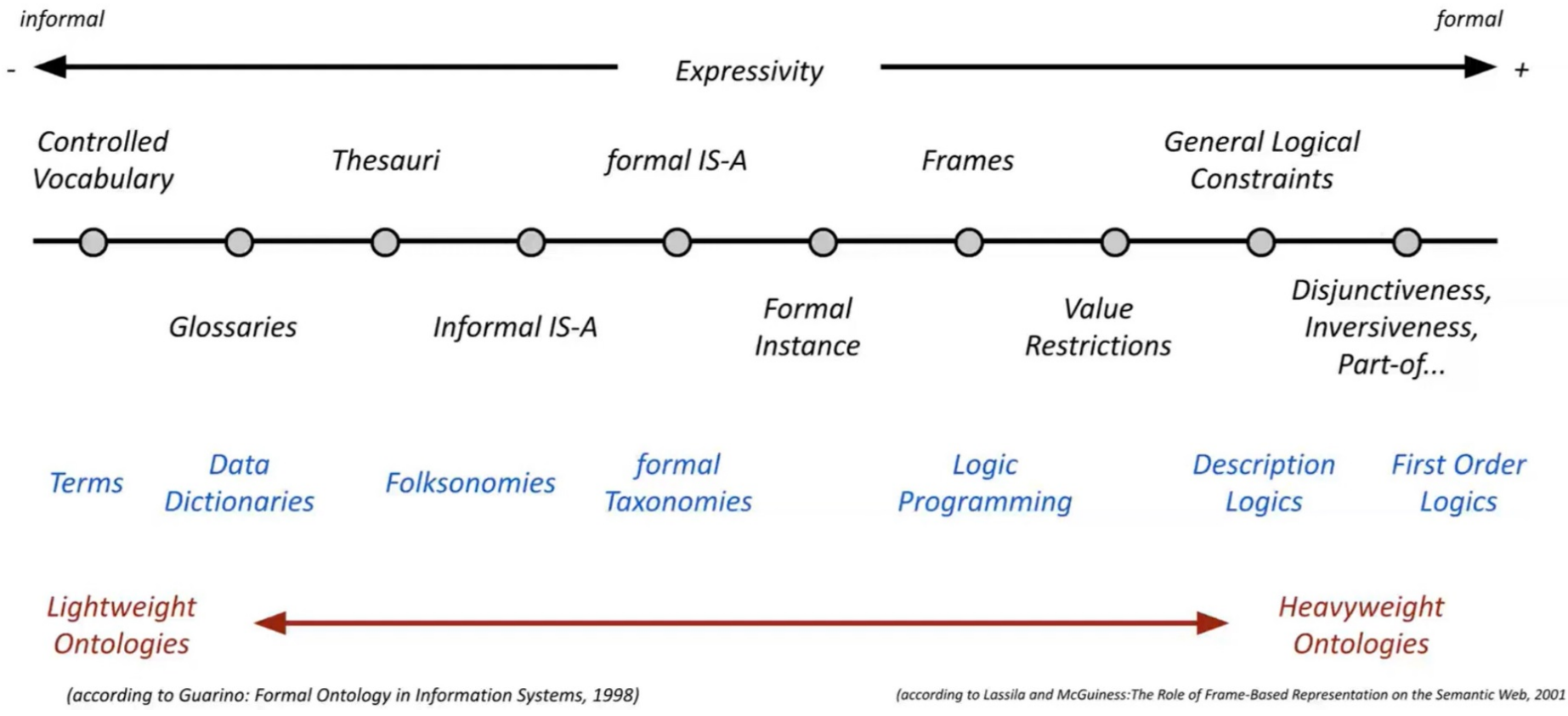
Interpretation als RDF Schema

- Wieder ein anderes Datenmodell
- `rdf:type` und `rdf:subClassOf` werden speziell interpretiert



Beschreibungslogiken

Übersicht Ontologie Typen und Kategorien



Ontologie Typen und Kategorien entsprechend semantischer Ausdrucksfähigkeit

Übersicht

Beschreibungslogiken (engl. Description Logics, *DLs*)

- Familie von Formalismen zur expliziten und impliziten Repräsentation von strukturiertem Wissen
- Logikbasiert
  - logikbasierte Semantik
  - automatische Deduktion

Vorteile

- Ausdrucksstark für komplexes Wissen
- Schlank genug für Anwendbarkeit

Anwendungsgebiete und Literaturempfehlung

Anwendungsgebiete von Beschreibungslogiken

- Kerntechnologie des Semantic Web
- Grundlage für Ontologiesprache OWL (Web Ontology Language, W3C Standard April 2004)
- Grundlage für semantisches Wissensmanagement (z.B. in Unternehmen)

Literaturempfehlung

- Baader et al., The Description Logic Handbook, Cambridge University Press, 2007

Youtube-Videoempfehlung

- Description Logics ALC, Open HPI, Dr. Harald Sack

Wichtige Inferenzprobleme als Anwendungsbeispiele

- Globale (In-)Konsistenz der Wissensbasis $KB \models \bot$?
  - ist die Wissensbasis sinnvoll?
- Klassen(in-)konsistenz $C \equiv \bot$?
  - Muss Konzept C leer sein?
- Klasseninklusion (Subsumption) $C \sqsubseteq D$?
  - Strukturierung der Wissensbasis
- Klassenäquivalenz $C \equiv D$?
  - Sind zwei Konzepte dieselben?
- Klassendisjunktheit $C \sqcap D = \bot$?
  - Sind zwei Konzepte disjunkt?
- Klassenzugehörigkeit $C(a)$?
  - Ist Individuum a in der Klasse C enthalten?
- Instanzgenerierung (Retrieval) “alle x mit $C(x)$ finden“?
  - Finde alle bekannten Individuen zum Konzept C.

Beschreibungslogiken und Prädikatenlogik 1. Stufe

Beschreibungslogiken (z.B. $\mathcal{ALC}$) sind Fragmente der **Prädikatenlogik 1. Stufe** (engl. First Order Logic, *FOL*).

Ein wichtiger Unterschied zur Prädikatenlogik ist jedoch, dass viele beschreibungslogische Sprachen **entscheidbar** sind.

Dies ermöglicht über eine Beschreibungslogik zu **schließen**.
$\Rightarrow$ **Implizites Wissen** kann durch Schlussfolgerung aus einer Wissensbasis **abgeleitet** werden.

Herausforderung und Zielsetzung

Herausforderung

Je ausdrucksstärker die Logik, desto schwieriger das automatische Schließen.

$\Rightarrow$ Kompromiss zwischen Ausdrucksstärke und Komplexität des Schlussfolgerns muss gefunden werden.

Zielsetzung

Gesundes Gleichgewicht finden, d.h. möglichst ausdrucksstarke Logik, die für wichtige Probleme entscheidbares und möglichst effizientes automatisches Schließen ermöglicht.

Konstruktoren

Familie von Beschreibungslogiken

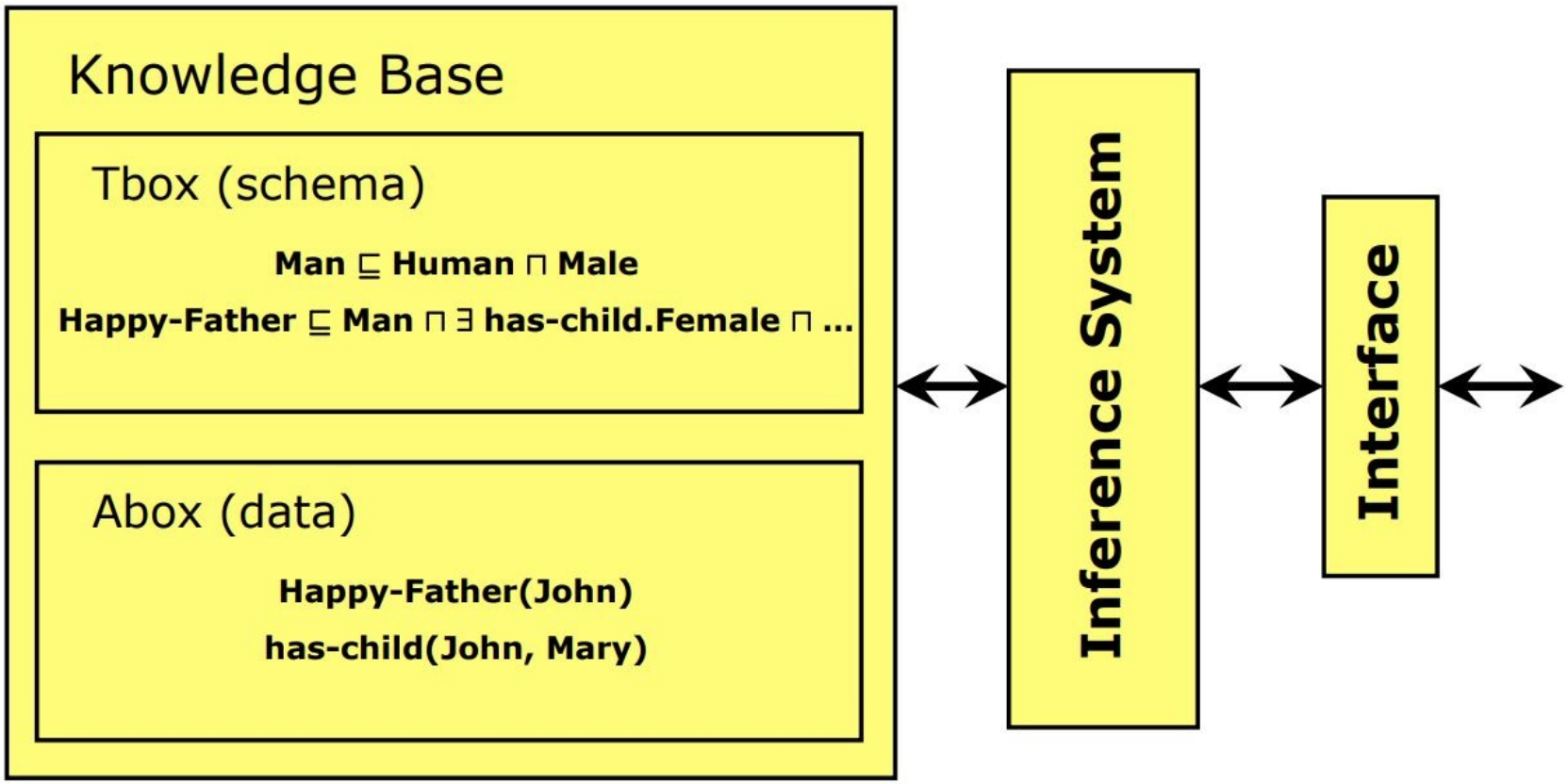
Es gibt nicht *die* eine Beschreibungslogik, sondern **viele verschiedene** Beschreibungslogiken.

Konstruktoren

- In Beschreibungslogiken werden mittels sogenannter **Konstruktoren** aus einfachen Beschreibungen komplexere aufgebaut.
- Verschiedene Beschreibungslogiken **unterscheiden sich in der Menge der Konstruktoren** (=Ausdrucksstärke), die sie enthalten.
- Konstruktoren ermöglichen den **Aufbau** von **komplexeren Konzepten aus** weniger komplexen bzw. **atomaren Konzepten.**
- Welche Arten von Konstruktoren es gibt, hängt von der konkreten DL ab.
- Die meisten Beschreibungslogiken bieten jedoch
  - Konjunktion $\sqcap$, Disjunktion $\sqcup$ und Negation $\neg$
  - Existenzquantor $\exists$ und Allquantor $\forall$

DL Architektur

Eine **Wissensbasis** in der Beschreibungslogik besteht aus einer **TBox** und einer **ABox:**



TBox und ABox

Terminologisches Wissen (TBox)

Axiome, die die Struktur der zu modellierenden Domäne beschreiben. Konzepte werden definiert und zueinander in Beziehung gebracht (konzeptionelles Schema).

- Aussagen der Form $C \sqsubseteq D$ und $C \equiv D$; C, D: komplexe Konzepte

Beispiele

- **Konzeptdefinition** $Orphan \equiv Human \sqcap \neg \exists has.Parent.Alive$
- **Allgemeines Hintergrundwissen bzw. Constraint** $Human \sqsubseteq \exists hasParent.Human$

Wissen um Individuen (ABox)

Axiome, die konkrete Situationen (Daten) beschreiben.

- Aussagen der Form $C(a)$ und $r(a, b)$, C komplexes Konzept, r Rolle und a, b Individuen
- **Beispiele:** $Orphan(harrypotter)$ $hasParent(harrypotter, jamespotter)$

Atomare Typen

Wir fixieren von nun an jeweils eine abzählbar unendliche Menge für **Konzeptnamen** *(auch: Klassennamen)*

- beschreiben Klassen von Objekten
- stellen einstellige Prädikate dar
- Beispiele: Person, Student, Hochschule, Vorlesung, Studiengang,...

und **Rollennamen**

- verbinden zwei Klassen oder Individuen
- stellen zweistellige Prädikate dar
- Beispiele: Student *besucht* Vorlesung, Mitarbeiter *betreut* Vorlesung,...

Wir nehmen die beiden Mengen als **disjunkt** an und unterscheiden Konzept- und Rollennamen über Groß- und Kleinschreibung.

ABox Syntax

Konzept Assertion C(a)

- Beispiel: $Student(Peter)$, $Vorlesung(KI)$
- Vergleichbar mit Objekten in UML und Entitäten in ER.

Rolle Assertion r(a, b)

- Beispiel: $besucht(Peter, KI)$
- Vergleichbar mit Assoziationen in UML und Beziehungen in ER.

Hinweise

- ABox ist eine **endliche Menge** solcher Axiome (Konzept Assertion und Rolle Assertion).
- Die in ABox benutzten Konzepte können, aber **müssen nicht** in TBox definiert sein.

$\mathcal{ALC}$: Basis-Beschreibungslogik

Attributive (Concept) Language with Complement ($\mathcal{ALC}$)

Einfachste Beschreibungslogik mit Booleschen Konstruktoren (and, or, not), die aussagenlogisch abgeschlossen ist.

Konstruktoren für $\mathcal{ALC}$

$\neg C$:	Negation
$C \sqcap D$:	Konjunktion
$C \sqcup D$:	Disjunktion
$\exists R.C$:	Existenzquantor
$\forall R.C$:	Allquantor

Atomare Typen

- Konzeptnamen A, B,... und zwei spezielle Konzepte:
  - $\top := A \sqcup \neg A$ *Top oder universelles Konzept*
  - $\bot := A \sqcap \neg A$ *Bottom Konzept*
- Rollennamen r, s,...

$\mathcal{ALC}$ Konstruktoren

Negation von C bedeutet intuitiv “alles außer C“

- $Mann \equiv \neg Frau$

Konjunktion von C und D bedeutet intuitiv “sowohl C als auch D“

- $Touchscreen \equiv Eingabegeraet \sqcap Ausgabegeraet$

Disjunktion von C und D bedeutet intuitiv “C oder D“

- $DHBWAngestellte \equiv Mitarbeiter \sqcup Professor$

Übersicht Beschreibungslogiken

Verschiedene **Konstruktormengen** ergeben verschiedene **Beschreibungslogiken:**

	Concepts	
$\mathcal{ALC}$	Atomic	A, B
	Not	¬C
	And	C ⊓ D
	Or	C ⊔ D
	Exists	∃ R.C
	For all	∀ R.C
$\mathcal{Q}$ ($\mathcal{N}$)	At least	≥n R.C (≥n R)
	At most	≤n R.C (≤n R)
$\mathcal{O}$	Nominal	$\{i_1, \ldots, i_n\}$

	Roles	
$\mathcal{I}$	Atomic	R
	Inverse	R^-

	Concept Axioms (TBox)	
	Subclass	C ⊑ D
	Equivalent	C ≡ D

	Role Axioms (RBox)	
$\mathcal{H}$	Subrole	R ⊑ S
$\mathcal{S}$	Transitivity	Trans(S)

	Assertional Axioms (ABox)	
	Instance	C(a)
	Role	R(a,b)
	Same	a = b
	Different	a ≠ b

- $\mathcal{S} = \mathcal{ALC}+$ Transitivität
- OWL DL $= \mathcal{SHOIN}(D)$ (D: Datentypen)
- OWL 2 $= \mathcal{SROIQ}(D)$

OWL

OWL - Allgemeines

Web Ontology Language (OWL)

- W3C Empfehlung seit 2004
- Semantisches Fragment der Prädikatenlogik erster Stufe
- OWL ist eine Familie von Sprachvarianten (engl. „*Species*“) mit verschiedenen Ausdrucksstärken
  - OWL Lite
  - OWL DL (entspricht Beschreibungslogik $\mathcal{SHOIN}(\mathcal{D})$)
  - OWL 2 (entspricht Beschreibungslogik $\mathcal{SROIQ}(\mathcal{D})$)
  - OWL Full
- keine Reifikation von “Aussagen über Aussagen“ in OWL DL
  $\rightsquigarrow$ RDFS ist Fragment von OWL Full

OWL DL stellt eine ausdrucksstarke Beschreibungslogik dar, die noch entscheidbar ist.

OWL 1 Varianten

OWL Full

- Enthält OWL DL und OWL Lite
- Enthält als einzige OWL-Teilsprache ganz RDFS
- Semantik enthält einige Aspekte, die aus logischem Blickwinkel problematisch sind
- Unentscheidbar
- Limitierte Unterstützung durch Softwaretools

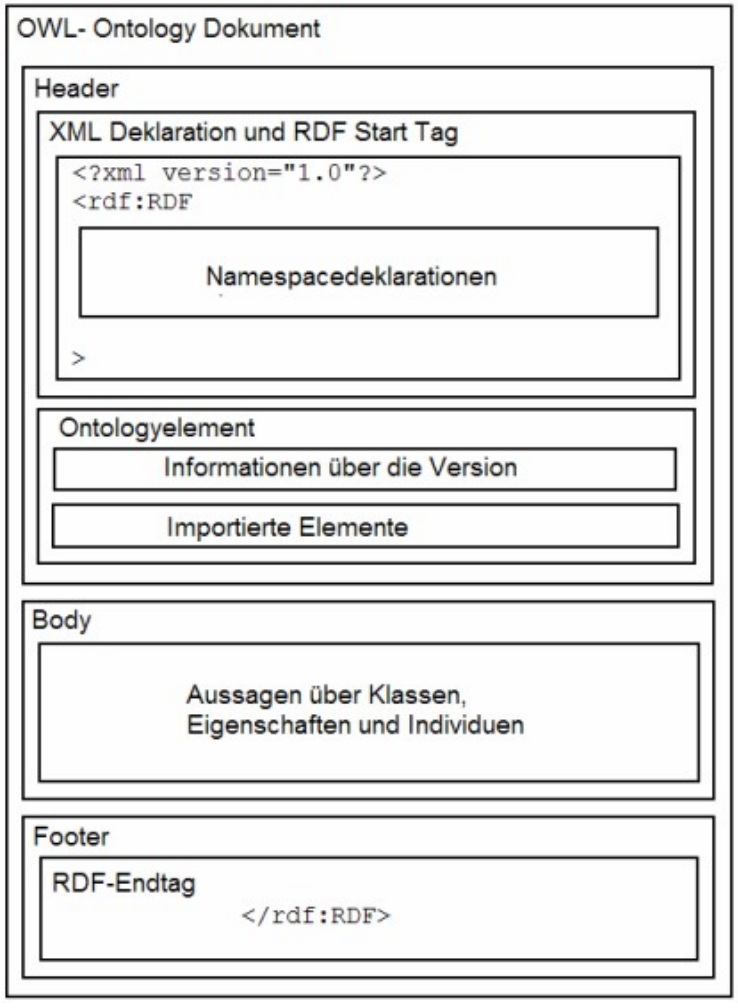
OWL DL

- Enthält OWL Lite und ist Teilsprache von OWL Full
- Vollständige Unterstützung durch Softwaretools
- Komplexitätsklasse `NEXPTIME` (worst-case)

OWL Lite

- Teilsprache von OWL DL und OWL Full
- Wenig ausdrucksstark
- Komplexitätsklasse `EXPTIME` (worst-case)

Aufbau von OWL Dokumenten



Aus: DB-Thüringen

OWL Dokumente

- sind RDF Dokumente (in der Standard-Syntax)
- bestehen aus
  - Kopf mit allgemeinen Angaben
  - Rest mit der eigentlichen Ontologie

Kopf eines OWL Dokumentes

Definition von Namespaces in der Wurzel

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

Kopf eines OWL Dokuments

Allgemeine Informationen

```
<owl:Ontology rdf:about="">
  <rdfs:comment
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string\>
    Beispiel Ontologie in der Version von Oktober 2011
  </rdfs:comment>
  <owl:versionInfo>v0.5</owl:versionInfo>
  <owl:imports rdf:resource="http://example.org/foo"/>
  <owl:priorVersion
    rdf:resource="http://example.org/projects/ex"/>
</owl:Ontology>
```

Beispiele

Von RDFS geerbt

rdfs:comment
rdfs:label
rdfs:seeAlso
rdfs:isDefinedBy

Für Versionierung

owl:versionInfo
owl:priorVersion
owl:backwardCompatibleWith

Außerdem

owl:imports

Klassen und Individuen

Klasse

- Definition
  - `<owl:Class rdf:ID="Professor"/>`
- Vordefinierte Klassen
  - `owl:Thing`
  - `owl:Nothing`

Individuum

- Definition durch Klassenzugehörigkeit

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

- Gleichbedeutend

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

Anfragen an OWL-Ontologien

Terminologische Anfragen an OWL

Anwendungsbeispiele für terminologische Anfragen an OWL
(nur Klassen und Rollen)

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

Beispiel einer inkonsistenten Klasse

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

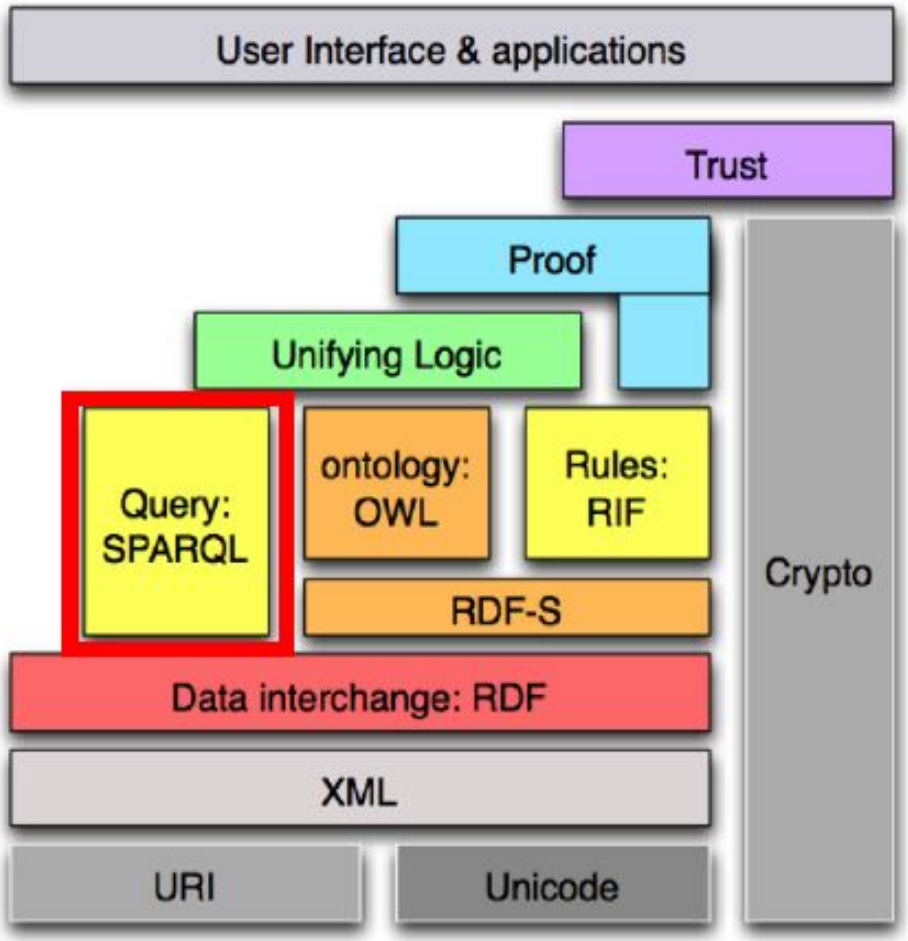
Assertionale Anfragen an OWL

Anwendungsbeispiele für assertionale Anfragen an OWL
(einschl. Individuen)

- Instanzüberprüfung: Gehört gegebenes Individuum zu gegebener Klasse?
- Suche nach allen Individuen, die in einer Klasse enthalten sind
- Werden zwei gegebene Individuen durch eine Rolle verknüpft?
- Suche nach allen Individuenpaaren, die durch eine Rolle verknüpft sind

Vorsicht: es wird nur nach “beweisbaren“ Antworten gesucht!

SPARQL


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

Anfragesprachen für das Semantic Web?

Wie kann auf in RDF oder OWL spezifizierte Informationen zugegriffen werden?

Abfrage von Informationen in RDF(S)

- Einfache Folgerung
- RDF-Folgerung
- RDFS-Folgerung

“Folgt ein bestimmter RDF-Graph aus einem gegebenen?“
“Folgt eine Subklassen-Beziehung aus einer OWL-Ontologie?“
“Welches sind die Instanzen einer Klasse einer OWL-Ontologie?“

Genügen OWL und RDF nicht?

Selbst OWL ist als Anfragesprache oft zu schwach

- "Welche Zeichenketten in deutscher Sprache sind in der Ontologie angegeben?"
- "Welche Properties verbinden zwei bestimmte Individuen?"
- "Welche Paare von Personen haben einen gemeinsamen Elternteil?"

$\rightarrow$ weder in RDF noch OWL ausdrückbar!

Anforderungen

- Große Ausdrucksstärke zur Beschreibung der gefragten Information
- Möglichkeiten zur Formatierung, Einschränkung und Manipulation der Ergebnisse

SPARQL

SPARQL

SPARQL Protocol And RDF Query Language

- sprich engl. „*sparkle*“
- Anfragesprache zur *Abfrage von Instanzen aus RDF-Dokumenten*
- Große praktische Bedeutung
- W3C Spezifikation SPARQL 1.1 seit 2013 offiziell empfohlen

Teile der SPARQL Spezifikation

- Anfragesprache
- Ergebnisformat: Darstellung von Ergebnissen in XML
- Anfrageprotokoll: Übermittlung von Anfragen und Ergebnissen

Einfache Anfrage

Eine einfache Beispielanfrage

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name .
        ?x foaf:mbox ?mbox }
```

- Die Bedingung der WHERE Klausel heißt **Query Pattern** *(Abfragemuster)*
- Tripel mit Variablen heißen **Basic Graph Pattern** *(BGP)*
  - BGPs verwenden Turtle Syntax für RDF
  - BGPs können Variablen (`?variable`) enthalten
- Abfrageergebnis für die **selektierten Variablen** (`SELECT`)

SPARQL Demo

Demo: Apache Jena Fuseki Server

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX sn: <http://www.snee.com/hr/>

SELECT ?givenName ?familyName ?hireDate
WHERE
{
   ?person vcard:given-name ?givenName .
   ?person vcard:family-name ?familyName .
   ?person sn:hireDate ?hireDate .
   ?person sn:completedOrientation ?completedOrientation .

}
```

QUERY RESULTS

Table | Raw Response

Showing 1 to 3 of 3 entries

Search: Show 50 entries

	givenName	familyName	hireDate
1	"Heidi"	"Smith"	"2015-01-13"
2	"John"	"Smith"	"2015-01-28"
3	"John"	"Smith"	"2015-01-28"

Showing 1 to 3 of 3 entries

```
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix sn: <http://www.snee.com/hr/> .

sn:emp1   vcard:given-name   "Heidi" .
sn:emp1   vcard:family-name   "Smith" .
sn:emp1   vcard:title   "CEO" .
sn:emp1   sn:hireDate   "2015-01-13" .
sn:emp1   sn:completedOrientation   "2015-01-30" .

sn:emp2   vcard:given-name   "John" .
sn:emp2   vcard:family-name   "Smith" .
sn:emp2   sn:hireDate   "2015-01-28" .
sn:emp2   vcard:title   "Engineer" .
sn:emp2   sn:completedOrientation   "2015-01-30" .
sn:emp2   sn:completedOrientation   "2015-03-15" .

sn:emp3   vcard:given-name   "Francis" .
sn:emp3   vcard:family-name   "Jones" .
sn:emp3   sn:hireDate   "2015-02-13" .
sn:emp3   vcard:title   "Vice President" .

sn:emp4   vcard:given-name   "Jane" .
sn:emp4   vcard:family-name   "Berger" .
sn:emp4   sn:hireDate   "2015-03-10" .
sn:emp4   vcard:title   "Sales" .
```

Quelle: http://www.learningsparql.com/

Lernkontrolle

Meine 3 Lernziele für heute waren

1. Ich kenne die grundlegenden Begriffe des Sematic Web und kann diese in einen Kontext setzen.
2. Ich bin vertraut im Umgang mit RDF und RDF Schema in einfachen Ontologien.
3. Durch die Anwendung grundlegender SPARQL-Syntax kann ich in OWL-Ontologien enthaltenes Wissen abfragen.