OWL
Web Ontology Language

Anne Schneider, Institut für Informatik, Friedrich-Schiller-Universität Jena

Übersicht

- 1. <u>Einführung, Motivation</u>
  - 1.1 Ontologien
  - 1.2 Das Semantic Web
  - 1.3 XML
  - 1.4 RDF und RDF/XML
    - 1.4.1 RDF
    - 1.4.2 RDF/XML
  - 1.5 RDF Schema
  - 1.6 Beschreibungslogiken

Übersicht

- 2. OWL
  - 2.1 Die Subsprachen
    - 2.1.1 OWL Full
    - 2.1.2 OWL DL
    - 2.1.3 OWL Lite
  - 2.2 Wie ist ein OWL Dokument aufgebaut?
    - 2.2.1 Klassen
    - 2.2.2 Eigenschaften
  - 2.3 OWL - Das Allheilmittel?

1.1 Das Semantic Web

- Erweiterung des WWW um maschinenlesbare Daten, welche die Semantik der Inhalte formal festlegen
- Konzept von Tim Burners Lee
- Programme sollen auf den Informationen operieren können

1.2 Ontologien

- aus dem Griechischen (Lehre des Seins)
- Eigentlicher Begriff aus der Philosophie
- bei Aristoteles: Studium der Wissensrepräsentation und des Schlussfolgerns
- in der Informatik im Bereich der Wissensrepräsentation ein **formal** definiertes System von Konzepten und Relationen
- enthalten Inferenz- und Integritätsregeln
- T. Gruber : *"explizite formale Spezifikation einer gemeinsamen Konzeptualisierung"*

1.2 Ontologien

Aufgaben/Merkmale

- gemeinsames Verständnis eines Fachbereichs kommunizieren
- explizite Semantik deklarieren
- ausdrucksstarke Aussagen treffen
- Informationen teilen

-> dazu werden Klassen, Eigenschaften und Individuen von Klassen festgelegt und semantische Beziehungen zwischen diesen:

  - Synonymie/Antonymie (=/≠)
  - Hyponymie (Generalisierung/Spezialisierung) (⊼)
  - Meronymie/Holonymie (◊)

1.3 XML

- Extensible Markup Language
- tagbasierte Metasprache
- für OWL genutzt, weil ...
  - weit verbreiteter Webstandard
  - XML jedem erlaubt Markupsprachen für eine bestimmte Domäne zu entwickeln
  - den Datenaustausch unterstützt
  - einfach zu parsen
  - anbieterneutrales Format

1.3 XML

- Elemente, Attribute und Inhalt werden durch benannte Markuptags definiert

```
<tagdescription> Inhalt </tagdescription>
```

- oder selbst schließende Tags

```
<tagname Inhalt />
```

- Inhalte können Texte sein oder andere Tags mit Inhalt
- Struktur durch Verschachtelung (Baumstruktur)

1.3 XML

Ein kurzes Beispiel:

```
<menu>
    <menuitem>
      <dishName>Taco con
      Pollo</dishName>
      <dishPrice>2,50€</dishName>
    </menuitem>
    <menuitem>
      <dishName>Taco Verdes</dishName>
      <dishPrice>1,50€</dishName>
    </menuitem>
    …
</menu>
```

Menu - Aj Caramba

Taco con Pollo	2,50
Taco con Porcino	2,50
Taco Verdes	1,50
Burrito Con Pollo	3,50
Burrito Verdes	2,50

…

1.3 XML

WARUM XML NICHT GENUG IST:

- XML definiert Syntax, aber nicht Semantik
- XML Beschreibungen sind für Computer mehrdeutig
- XML Tags sind nicht besser als natürliche Sprache (Strings, bedeutungsvoll für Menschen, nicht aber für Software)
- es lassen sich keine Relationen ausdrücken

1.4.1 RDF

- Resource Description Framework
- Datenmodell zur Bereitstellung von Metadaten im WWW
- maschinenverarbeitbare Beschreibungen, die einheitlich, formal, und standardisiert sind

Aussagen:

- Beschreiben Ressourcen und deren Attribute
- sind Tripel (Ressource, Eigenschaft, Literal oder Ressource)

	Subjekt	Prädikat	Objekt
Literalbewertete Aussage	Taco con Pollo	Preis	2,50€
Ressourcenbewertete Aussage	Paolo Rodriguez	Lieblingsessen	Taco con Pollo

1.4.1 RDF

- Warum auch RDF nicht genug ist:
  - gebraucht wird noch ein Format um die Inhalte vom RDF Datenmodell auszutauschen
  - Semantik ist noch nicht komplex genug um Ontologien zu beschreiben

→RDF/XML

- XML-Syntax für das RDF Datenmodell

Ein kurzes Beispiel:

```
<rdf:Description rdf:about="#PaoloRodriguez">
   <profile:favouriteDish>
    Taco con Pollo
   </profile:favouriteDish>
</rdf:Description>
```

1.4.2 RDF/XML

Warum selbst RDF/XML nicht reicht:

- noch nicht komplex genug
- keine Beschreibung von Klassen möglich
- kein Konzept für Aufzählungen und Datentypen (abgesehen von den typisierten Literalen)

1.5 RDF Schema

- Domänenneutrale, formale Schema Sprache
- stellt Basisstrukturen für Klassen und deren Eigenschaften (Properties) zur Verfügung
- erweitert das Typenkonzept von RDF
- unterstützt Einschränkungen von Klassen und Eigenschaften

RDFS Klassen

- haben assoziierte Ressourcen (Instanzen der Klasse)
- `rdfs:Class` um Klassen zu definieren
- `rdf:type` um Mitgliedschaft an Klassen zu definieren

1.5 RDF Schema

Vordefinierte Klassen:

- RDFS Ressource `rdfs:Resource`
- RDF Eigenschaften `rdf:Property`
- RDFS Klassen `rdfs:Class`
- RDFS Literale `rdfs:Literal`
- XML Literale `rdf:XMLLiteral`
- RDFS Datentypenklasse `rdfs:Datatype`

1.5 RDF Schema

Eigenschaften (Properties)

- können Beschränkungen enthalten

Kerneigenschaften

- `rdf:type`
- `rdfs:subClassOf`
- `rdfs:subPropertyOf`
- `rdf:domain/rdf:range`

1.5 RDF Schema

Warum RDFS noch immer nicht genug ist:

- keine Aussagen über Kardinalitäten
- keine Aussagen über Eigenschaften von Eigenschaften
- keine Aussagen über Äquivalenz, Schnitt, Vereinigung etc. von Objekten
- benötigt wird noch umfangreicheres Vokabular
- noch zu wenige Möglichkeiten um Eigenschaften einzuschränken
- zu wenige Deskriptoren um Inferenzierung zu unterstützen

1.6 Beschreibungslogiken

- engl. description logic (DL)
- Formalismus, der klassenbasiert Wissen repräsentiert

- DL Wissensbasis besteht aus Axiomen
- Domäne ($\Delta^{\mathcal{I}}$):

  Menge von Objekten
- Funktion ($\bullet^{\mathcal{I}}$):

  Abbildung von Individuen, Klassen und Eigenschaften auf Elemente, Teilmenge oder binäre Relation auf der Domäne

1.6 Beschreibungslogiken

Konstruktor	Syntax	Semantik		
Atomares Konzept	A	$A^I \subseteq \Delta^I$	$\mathcal{AL}$	$\mathcal{S}$
Atomare Rolle	R	$R^I \subseteq \Delta^I \times \Delta^I$		
Transitive Rolle	$R \in R_+$	$R^I = (R^I)^+$		
Konzeptkonjunktion	$C \sqcap D$	$C^I \cap D^I$		
Allquantor	$\forall R.C$	$\{a \in \Delta^I \mid \forall b.\langle a,b\rangle \in R^I \Rightarrow b \in C^I\}$		
Negation	$\neg C$	$\Delta^I \setminus C^I$	$\mathcal{C}$	
Existenzquantor	$\exists R.C$	$\{a \in \Delta^I \mid \exists b.\langle a,b\rangle \in R^I \wedge b \in C^I\}$	$\mathcal{E}$	
Konzeptdisjunktion	$C \sqcup D$	$C^I \cup D^I$	$\mathcal{U}$	
Rollenhierarchie	S	$R^I \subseteq S^I$	$\mathcal{H}$	
Inverse Rollen	R^-	$\{\langle a,b\rangle \mid \langle b,a\rangle \in R^I\}$	$\mathcal{I}$	
Zahlenrestriktion	$\geq nR$	$\{a \mid \#\{b.\langle a,b\rangle \in R^I\} \geq n\}$	$\mathcal{N}$	
	$\leq nR$	$\{a \mid \#\{b.\langle a,b\rangle \in R^I\} \leq n\}$		
Qualifizierte	$\geq nR.C$	$\{a \mid \#\{b.\langle a,b\rangle \in R^I \wedge b \in C^I\} \geq n\}$	$\mathcal{Q}$	
Zahlenrestriktion	$\leq nR.C$	$\{a \mid \#\{b.\langle a,b\rangle \in R^I \wedge b \in C^I\} \leq n\}$		

2 OWL- Eine Einführung

- Web Ontology Language
- Standardsprache um im Semantic Web Informationen zu repräsentieren
- zur Beschreibung von Ontologien
- Menge von Axiomen die Klassen und Eigenschaften (Properties), sowie die Beziehungen zwischen diesen, beschreiben

2 OWL- Eine Einführung

Geschichte

- **DARPA Agent Markup Language (DAML)** 2000 entwickelt von der DARPA um Interoperabilität zwischen den IT Systemen des Department of Defence zu ermöglichen
- **Ontology Inference Layer (OIL) Language** parallel von EU-Projekt entwickelt
- März 2001 ein Komitee aus der EU und den US mischt DAML und OIL zu **DAML+OIL**
- Web Ontology (WebOnt) Working Group (Leitung Dr. James Hendler und Dr. Guus Schreiber) entwickeln OWL aus DAML+OIL
- Februar 2006 erfolgreich durch W3C standardisiert

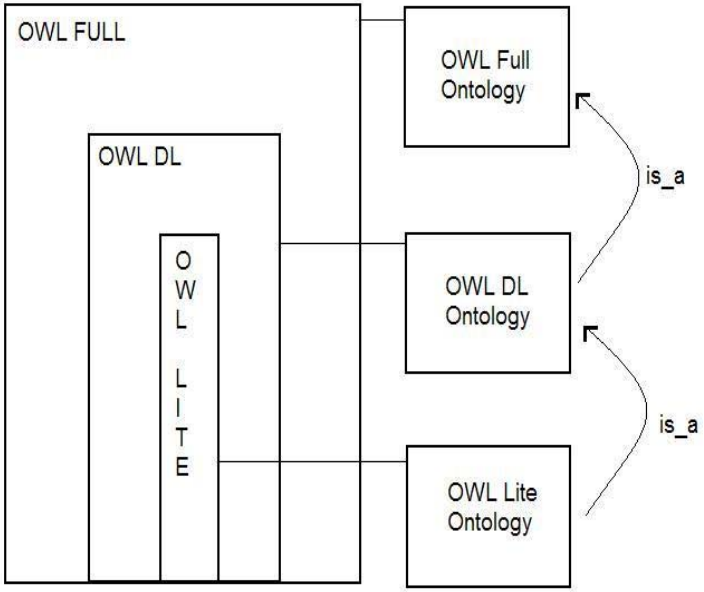
2 OWL- Eine Einführung

OWL kann:

- Klassen logisch kombinieren:
  - Vereinigung
  - Durchschnitt
  - Komplement

- Eigenschaften definieren als:
  - transitiv
  - symmetrisch
  - funktional
  - als inverses einer anderen Eigenschaft

2.1 Die Subsprachen



um unterschiedlichen Nutzergruppen gerecht werden zu können:

- Berechenbarkeit
- Expressivität
- Erlernbarkeit für den Menschen

2.1.1 OWL Full

- umfasst alle OWL Sprachkonstrukte
- Obermenge von RDF
- RDF Aussagen können mit OWL Full Konstrukten beliebig gemischt werden
- Problem: nicht entscheidbar

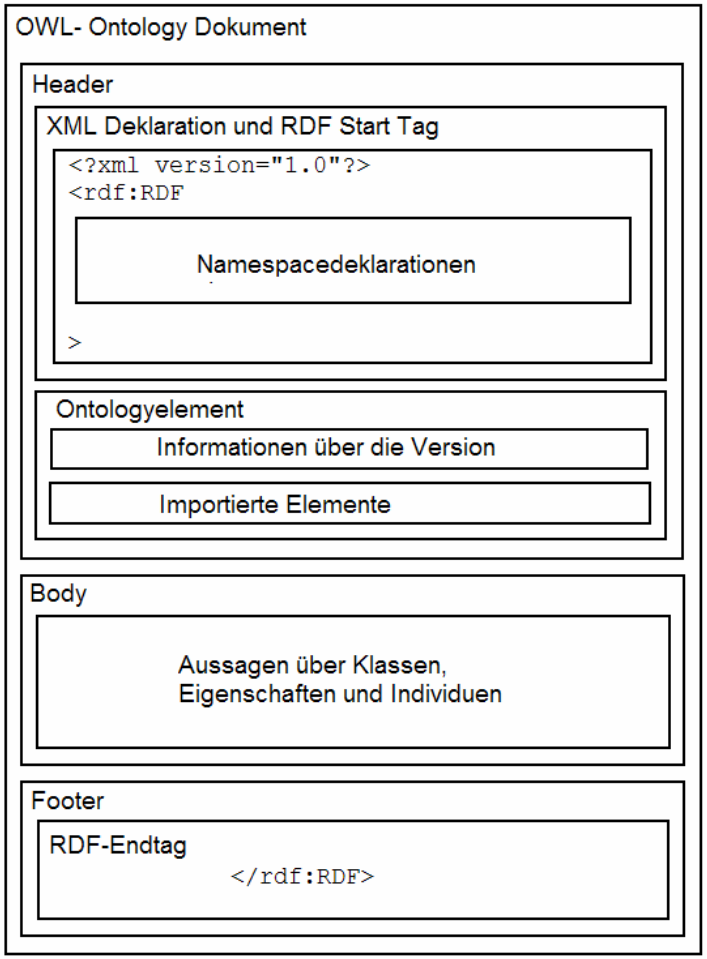
2.1.2 OWL DL

- $\mathcal{SHOIN}(\mathbf{D})$
- DL steht für Description Language
- Subsprache von OWL Full
- Kompromiss zwischen Ausdruckskraft und Entscheidbarkeit
- teilweise sind OWL Full Sprachkonstrukte im Gebrauch eingeschränkt

2.1.3 OWL Lite

- $\mathcal{SHIF}(\mathrm{D})$
- Subsprache von OWL DL
- zur Erstellung einer einfachen Klassenhierarchie
- Konstrukte, die nicht genutzt werden:
  - `owl:oneOf`
  - `owl:unionOf`
  - `owl:complementOf`
  - `owl:hasValue`
  - `owl:disjointWith`
  - `owl:DataRange`
- manche Konstrukte unterliegen Restriktionen
- Auswertung der Ausdrücke leichter

2.2 Wie ist ein OWL Dokument aufgebaut?



- Axiome mit Klassen und Eigenschaften
- Klassen:

  Menge von Individuen, die bestimmte Properties (Eigenschaften) teilen
- Eigenschaften:

  Beziehungen zwischen Ressourcen und Ressourcen oder Ressourcen und Datentypwerten
- Reihenfolge der Axiome ist egal (Forward-Referenzierung ist möglich)

2.2.1 Klassen

- werden definiert durch:
  - Namensgebung
  - Spezifizerung
- Instanzen von `owl:Class` (ist eine Subklasse von `rdf:Class`)

- `<owl:Class rdf:ID=„dishes"/>`

- Es gibt zwei vordefinierte Klassen
  - `owl:Thing` Oberklasse aller Klassen
  - `owl:Nothing` leere Klasse (Unterklasse aller Klassen)

2.2.1 Klassen

- prinzipieller Aufbau von Restriktionen:

```
<owl:Class rdf:ID=„Klasse">
  <owl:equivalentClass>
   <owl:Restriction>
   <owl:onProperty rdf:resource=„Property“/>
      value constraint/cardinality constraint
   </owl:Restriction>
  </owl:equivalentClass>
</owl:class>
```

2.2.1 Klassen

Wertbeschränkungen- Quantoren:

- Allquantor (∀)
  Alle Property Werte müssen zur spezifizierten Klasse gehören

```
<owl:Restriction>
   <owl:onProperty rdf:resource="#CoveredWith" />
   <owl:allValuesFrom rdf:resource=„#Salsa" />
</owl:Restriction>
```

- Existenzquantor (∃)
  mindestens ein Property-Werte müssen zur spezifizierten Klasse gehören

```
<owl:Restriction>
   <owl:onProperty rdf:resource="#contains" />
   <owl:someValuesFrom rdf:resource=„#beans" />
</owl:Restriction>
```

2.2.1 Klassen

Wertbeschränkung

Klasse basiert auf der Existenz eines bestimmten Property-Wertes

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#favouriteMeal"/>
  <owl:hasValue rdf:resource="#TacoConPollo" />
</owl:Restriction>
```

- Nicht in OWL Lite

2.2.1 Klassen

Kardinalitätsbeschränkung:

```
<owl:Restriction>
   <owl:onProperty rdf:resource="#hasPrice"/>
   <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
    1
   </owl:minCardinality>
</owl:Restriction>
```

- auch als `maxCardinality` oder `Cardinality`
- in OWL Lite nur 0 oder 1

2.2.1 Klassen

Aufzählungen:

Klasse wird beschrieben indem ihre Mitglieder aufgezählt werden

```
<owl:Class rdf:ID="menu">
   <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#TacoConPollo"/>
    <owl:Thing rdf:about="#TacoConPorcino"/>
    <owl:Thing rdf:about="#TacoVerde"/>
    <owl:Thing rdf:about="#BurritoConPollo"/>
    <owl:Thing rdf:about="#BurritoVerde"/>
   </owl:oneOf>
</owl:Class>
```

- nicht in OWL Lite
- auch erweiterbar

2.2.1 Klassen

Vereinigung:

definiert Klasse, die alle Mitglieder aller spezifizierten Klassen enthält

```
<owl:Class rdf:ID="MeatDishes">
  <owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#DishesWithChicken"/>
   <owl:Class rdf:about="#DishesWithPork"/>
  </owl:unionOf>
</owl:Class>
```

- nicht in OWL Lite

2.2.1 Klassen

Durchschnitt:

definiert Klasse, die alle Mitglieder enthält die zu den spezifizierten Klassen gleichzeitig gehören

```
<owl:Class rdf:ID="MeatDishes">
    <owl:intersectionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Dishes"/>
     <owl:Restriction>
          <owl:onProperty rdf:resource="#isMadeOf"/>
          <owl:hasValue rdf:resource="#Meat"/>
     </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

- in OWL Lite nur Durchschnitt von Klassennamen und Restriktionen (z.B. nicht möglich: Durchschnitt einer Vereinigung)

2.2.1 Klassen

Komplement:

definiert Klasse, indem alle Individuen, die nicht zu der Klasse gehören benannt werden

```
<owl:Class rdf:ID="NotADish">
   <owl:complementOf rdf:resource="#Dish"/>
</owl:Class>
```

- nicht in OWL Lite

Subklasse:

Spezialisierung einer anderen Klasse

```
<owl:Class rdf:ID="Dish">
   <rdfs:subClassOf rdf:resource="#DeliciousDish"/>
</owl:Class>
```

2.2.1 Klassen

Äquivalenzklasse:

die Klassen enthalten die gleichen Individuen

```
<owl:Class rdf:about="#Dish">
  <owl:equivalentClass rdf:resource="#Meal"/>
</owl:Class>
```

die Klassen enthalten die gleichen Individuen

Disjunkte Klassen:

Individuen sind entweder in der einen oder in der anderen Klasse

```
<owl:Class rdf:about="#Meatdish">
  <owl:disjointWith rdf:resource="#VegeterianDish"/>
</owl:Class>
```

2.2.2 Eigenschaften

- Subklassen von `rdf:Property`
- Sollen Individuen einem Datentyp oder einem anderen Individuum zuordnen

<u>Vier disjunkte Arten</u>

- Objekteigenschaft

  ordnen Objekten Objekte zu

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

- Datentypeigenschaft

  ordnen Objekten Datentypen zu

  ```
  <owl:DatatypeProperty rdf:ID=„RestaurantClosingTime"/>
  ```

2.2.2 Eigenschaften

- Annotationseigenschaft
  - Beschreiben Objekte

```
<owl:AnnotationProperty rdf:ID=„uriRef"/>
```

  - In OWL DL eingeschränkt
  - Vordefinierte Annotationen sind `owl:versionInfo`, `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, `rdfs:isDefinedBy`

- Ontologie-Eigenschaft
  - Ordnen Ontologien Ontologien zu
  - Vordefinierte Properties sind `owl:Imports`, `owl:backwardCompatibleWith`, `owl:incompatibleWith`, `owl:priorVersion`

2.2.2 Eigenschaften

Charakteristiken von Eigenschaften

- aus RDF bereits bekannt:
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
- neu:
  - owl:equivalentProperty
  - owl:inverseOf
- ein Beispiel:

```
<owl:ObjectProperty rdf:ID=„menuPrice">
  <owl:equivalentProperty rdf:resource="#menuCost"/>
</owl:ObjectProperty>
```

2.2.2 Eigenschaften

- Transitivität:

  `<owl:TransitiveProperty>`

- Symmetrie:

  `<owl:SymmetricProperty>`

- Rechtseindeutigkeit:

  `<owl:FunctionalProperty>`

- Linkseindeutigkeit/Injektivität:

  `<owl:InverseFunctionalProperty>`

2.2.3 Fakten

- Neben den Klassen- und Eigenschaftsaxiomen gibt es noch individuelle Axiome –>Fakten

- Es gibt drei Arten:
  - Klassenzugehörigkeit
  - Eigenschaftswerte
  - Identitätsaussagen

2.2.3 Fakten

Klassenzugehörigkeit

```
<Dish rdf:ID=„TacoConPollo">
  <consistsOf rdf:resource="#Taco"/>
  <prize rdf:resource="#1,50€"/>
  …
  …
</Dish>
```

2.2.3 Fakten

- Identitätsaussagen
  - owl:sameAs
  - owl:differentFrom
  - owl:AllDifferent

- Ein Beispiel:

```
<Dish rdf:ID=„Meatdish">
 <owl:differentFrom rdf:resource="#TacoConPollo“/>
 <owl:differentFrom rdf:resource="#TacoConPorcino"/>
</Dish>
```

2.3 OWL - Das Allheilmittel?

- dient nur zur Repräsentation von Wissen
- erfüllt noch keinen alltäglichen Zweck
- gewaltig viel Arbeit vor uns um alles Wissen in Ontologien zu pressen
- Noch immer nicht ausdrucksstark genug
  - Wir können keine Folgerungen im Sinne der Prädikatenlogik machen (brauchen dazu einen Reasoner)
- ist nicht prozedural
  - schwer zu erweitern