

OWL und Beschreibungslogiken

Jens Wächter

Universität Hamburg - Fachbereich Informatik

Vorgehen

- OWL - Hintergrund, Einflüsse, Syntax, Struktur
- Ziel : basales Verständnis von OWL.
- Themenreihenfolge : DLs - RDF - Frames Paradigm - OWL
- Quellen : Ian Horrocks, Peter F. Patel-Schneider & Frank van Harmelen: From SHIQ and RDF to OWL: the making of a Web Ontology Language.
Resource Description Framework (RDF) : Concepts and Abstract Syntax
RDF Vocabulary Description Language 1.0: RDF Schema
OWL Web Ontology Language Guide en.wikipedia.org
Skript zur Einführung in die Beschreibungslogiken, Uni Kiel

Hintergrund

- OWL (“Web Ontology Language“) ist eine von der “Web Ontology Working Group“ entworfene Sprache zur Beschreibung von Ontologien.
- “The OWL Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications“ (W3C-OWL-Guide)
- OWL orientiert sich syntaktisch an anderen W3C Standards (XML (Schema), RDF (Schema))
- Es wurde versucht, Kompatibilität zu den bereits bestehenden Sprachen zur Beschreibung von Ontologien im Web “SHOE“, “OIL“ und “DAML-OIL“ zu halten ⇒ *Kompromisse*

Hintergrund(2)

- OWL besitzt drei in der Mächtigkeit wachsende Dialekte
OWL-Lite, OWL-DL und OWL-Full
- Eine Ontologie wird als Information über Kategorien von Objekten und die Beziehungen zwischen den Objekten aufgefasst. Es sind auch alternative Interpretationen des Ontologiebegriffs möglich.
- Es gibt drei Ebenen, OWL zu repräsentieren :
“Abstrakte Syntax“, RDF/XML und RDF Graphen.

Description Logics - Übersicht

- Beschreibungslogiken sind eine Familie von konzeptbasierten Formalismen zur Wissensrepräsentation. Sie besitzen eine abstrakte Syntax.
- Was in OWL "Ontologie" genannt wird, ist in DLs eine "knowledge base".
- Charakteristisch für DLs ist die Benutzung von Konstruktoren, um aus einfachen Klassen komplexere zu bauen, und der Bereitstellung von korrekten, kompletten und lenkbaren Diensten zum Rasonieren(Inferenz von nicht explizitem Wissen).
- DLs bieten als Logiken auch die Möglichkeit, eine Semantik (über die Modelltheorie) der Logik festzulegen

Description Logics - Übersicht (2)

- (Aus Wikipedia) : Unterscheidung von “TBox“ (terminological box) und “ABox“ (assertional box)
- ABox enthält Terme, die “ground“ sind. Dies sind Informationen über Individuen.
- TBox enthält Beschreibungen von Konzepthierarchien.

DLs - Semantik

- Ein Modell ist ein Tupel aus einer Domäne und einer Interpretationsfunktion : $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$.
- Die Domäne ist eine Menge von Objekten und die Interpretationsfunktion eine Abbildung von Individuen-, Klassen- und Eigenschaftsnamen auf die Elemente, Teilmengen der Domäne und binären Relationen über der Domäne.
- Die Interpretationsfunktion lässt sich mit den gängigen Operationen auf Mengen von den Namen von Klassen auf komplexe (zusammengesetzte) Klassen erweitern.

DLs - Semantik(2)

- Axiome in der Wissensbasis von DLs legen fest, wie die Beziehungen zwischen einzelnen Klassen zu verstehen sind (z.B. “subclass“ Relation).
- Diese Axiome werden in den Modellen zu “constraints“, z.B. in der “subclass“ Relation muss eine Klasse A, die Unterklasse von B ist, immer als Teilmenge von B aufgefasst werden.
- Beispiel : Das Axiom “Person \sqsubseteq Animal“ führt dazu, dass $Person^{\mathcal{I}} \subseteq Animal^{\mathcal{I}}$
- In OWL wird eine Modelltheorie vom Stile einer DL benutzt, um die Semantik festzulegen

DLs - Sprachkonstrukte

- Die Auswahl an Sprachkonstrukten bestimmt die Mächtigkeit der Sprache.
- DLs besitzen eine im Normalfall entscheidbare “decision procedure“, die testet, ob eine Ontologie aus einer anderen folgt.
- Sprachkonstrukte können auf Klassen- und “Property“-Ebene bestehen.
- Mögliche Sprachkonstrukte sind boolesche Verknüpfungen, Restriktionen auf den Wertebereich von Rollen, inverse und transitive Rollen

DLs - Datentypen

- In einigen DLs gibt es eine strikte Trennung zwischen abstrakten Klassen und “technischen“ Datentypen (und Werten).
- Dies erlaubt eine getrennte Interpretation (Δ_D^I).
- Die Definition der binären Relationen muss angepasst werden.
- Die Interpretation von Datentypen erfolgt durch ein Orakel, was Aussagen über Integer wie z.B. “1 < 2“ treffen kann.

ALC als Beispiel DL

Frau \equiv Mensch \sqcap Weiblich

Mann \equiv Mensch \sqcap \neg Weiblich

Mutter \equiv Frau \sqcap \exists hat-kind.Mensch

Vater \equiv Mann \sqcap \exists hat-kind.Mensch

Eltern \equiv Mutter \sqcup Vater

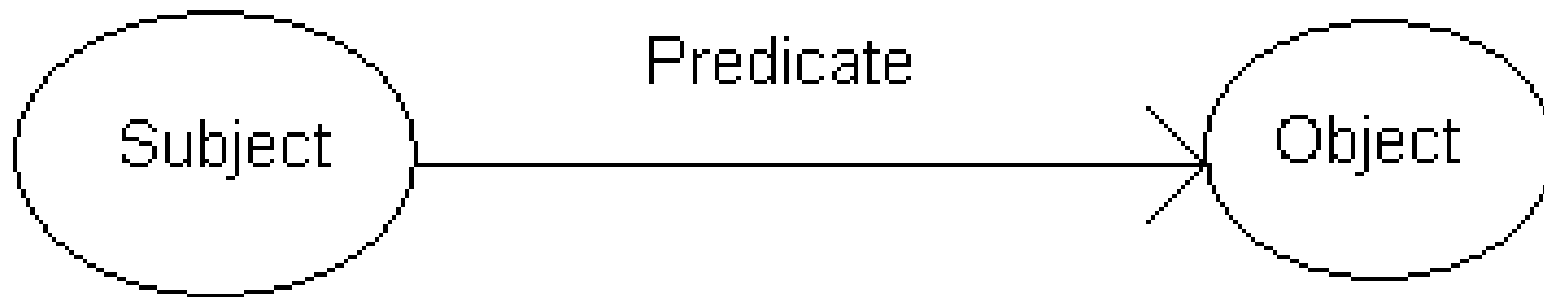
\forall nT \equiv Vater \sqcap \forall hat-kind.Frau

(Aus

Skript zur Einführung in die Beschreibungslogiken, Uni Kiel)

RDF(Link)

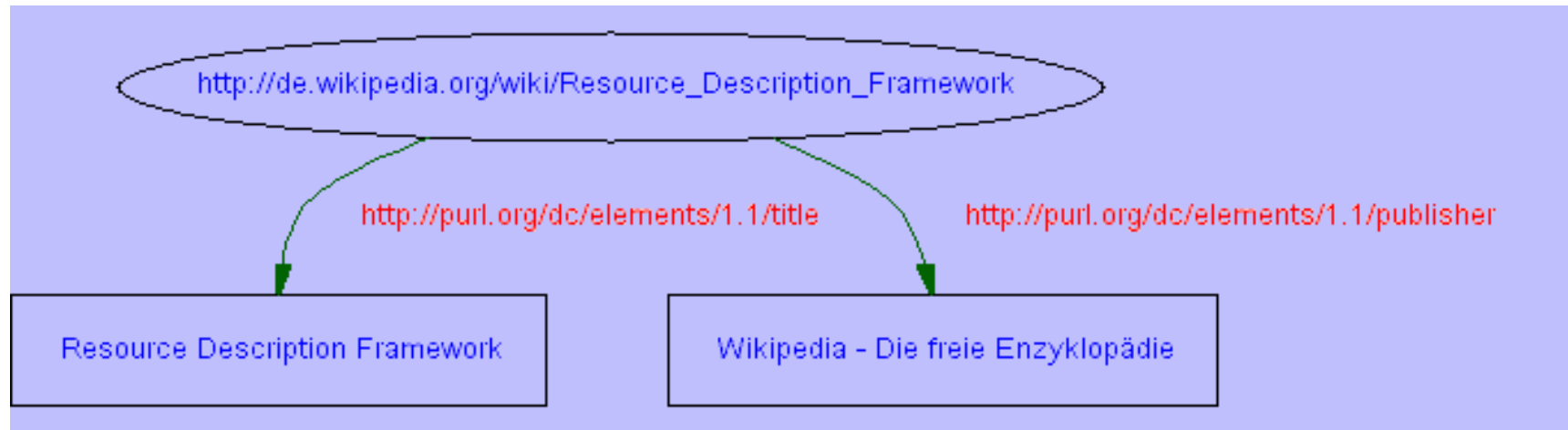
- RDF (Resource Description Framework) dient zur Darstellung von Informationen und besitzt im Gegensatz zu XML eine formale Semantik.
- Kern von RDF ist ein Graphen repräsentierendes Datenmodell :



RDF(2)

- der Graph wird als Tripel dargestellt : Subject Predicate Object
- Knoten sind entweder URIs, Literale(Strings, Integer) oder "blank nodes". "blank nodes" können unterschieden werden.
- ```
_:c owl:intersectionOf _:l1
_:l1 rdf:first ex:Student
_:l1 rdf:rest _:l2 .
_:l2 rdf:first ex:Employee
_:l2 rdf:rest rdf:nil .
```

# RDF-Beispiel



Quelle : [de.wikipedia.de/wiki/RDF](http://de.wikipedia.de/wiki/RDF)

# RDF Schema(Link)

- RDFS ist eine semantische Erweiterung zu RDF. Mit dieser Erweiterung lassen sich Gruppen von “Ressourcen“, die “Properties“ besitzen, und ihre Beziehungen beschreiben.
- “rdfs:Resource“ ist die Oberklasse aller andere Klassen. Weitere Klassen sind “rdfs:class“, “rdfs:datatype“ und andere
- “Properties“ wie z.B. “rdfs:subClassOf“ sind eine eigene Klasse und legen in der Tripel Notation (Subject-Predicate-Object) Relationen zwischen der Subject und Object “Resource“ fest.
- “C1 rdfs:subClassOf C2“ bedeutet, dass C1 und C2 Instanzen von “rdfs:class“ und C1 eine Unterklasse von C2 ist.

# RDF im XML-Syntax

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

(RDF/XML Beschreibung der RDF Klasse "class").

# Frames Paradigm

- Das Frames Paradigm spezifiziert eine Darstellungsform von Klassen und soll der Lesbarkeit dienen
- Es wird u.a. in Protégé genutzt
- Jede Klasse wird in einem Frame (bestehend aus Name, Klassen, aus denen die Klasse spezialisiert wird und “slots“) beschrieben.
- Frames sind semantisch äquivalent zu Axiomen von “subclass“ Beziehungen.

# Probleme mit Syntax und Semantik von RDF

RDF/XML ist "laut" :

Student = Person  $\sqcap \geq 1$  enrolledIn wird in RDF/XML zu

```
<owl:Class rdf:ID="Student">
```

```
 <owl:intersectionOf rdf:parsetype="Collection">
```

```
 <owl:Class rdfs:about="Person"/>
```

```
 <owl:Restriction>
```

```
 <owl:onProperty rdf:resource="enrolledIn" />
```

```
 <owl:minCardinality rdfs:datatype="&xsd;Integer">
```

```
 1
```

```
 </owl:minCardinality>
```

```
 </owl:Restriction>
```

```
 </owl:intersectionOf>
```

```
</owl:Class>
```

# Probleme mit Syntax und Semantik von RDF

- In RDF wird jeder Ausdruck in Folge von Tripeln aufgebrochen. Wenn ein Tripel, das nicht aus einem Ausdruck der Sprache entstanden ist, angetroffen wird, was soll passieren?
- Die Tripel eines Ausdrucks müssen nicht notwendigerweise zusammen aufzufinden sein.
- RDF Tripel können zirkulär sein :  
\_:x owl:onProperty ex:child  
\_:x owl:allValuesFrom \_:x

# OWL(1)

- Besitzt eine abstrakte Syntax wie eine DL.  
Class(Student complete  
Person  
restriction(enrolledIn minCardinality(1))).
- Im Gegensatz zu DLs benutzen die OWL Sprachen URI als Namen, so wie dies in RDF geschieht.
- Erweitert RDFS(OWL Full) bzw. ist größtenteils zu RDFS kompatibel.
- Sprachkonstrukte sind in OWL DL eingeschränkter als in OWL Full und in OWL Lite eingeschränkter als in OWL DL

# OWL(2)

- Die Klassen werden zu “Ontologien“ zusammengefasst und in RDF/XML gespeichert.
- Es lassen sich Ontologien aus dem Web importieren.
- OWL DL und OWL Lite haben eine an Frames orientierten abstrakte Syntax

# OWL DL

- Besitzt Konstrukte zur Beschreibung von Klassen, Datentypen Individuen und Datenwerten und benutzt diese um Axiome über Klassen, Eigenschaften und Individuen zu formulieren.
- Die Datentypen stammen aus RDF und XML-Schema.
- Hauptunterschied zu DLs ist die Verwendung von URIs und der XML-Schema Datentypen.

# OWL Lite

- Echte Teilmenge von OWL DL
- Kennt keine Vereinigung, Komplementbildung, auf Frame ähnliche Klassenaxiome beschränkte implizite Durchschnittsbildung, erlaubt nicht Individuen in der Beschreibung von Klassenaxiomen aufzutauchen und kennt nur die Kardinalitäten 0 und 1.

# OWL Full

- Kompatibel zu RDF(S)
- Keine abstrakte Syntax  $\Rightarrow$  RDF/XML
- Unentscheidbar
- Semantik aus der Modelltheorie

# OWL-DL Syntax

| Abstract Syntax                         | DL Syntax        | Semantics                                                                                     |
|-----------------------------------------|------------------|-----------------------------------------------------------------------------------------------|
| Descriptions ( $C$ )                    |                  |                                                                                               |
| $A$ (URI reference)                     | $A$              | $A^I \subseteq \Delta^I$                                                                      |
| owl:Thing                               | $\top$           | owl:Thing <sup>I</sup> = $\Delta^I$                                                           |
| owl:Nothing                             | $\perp$          | owl:Nothing <sup>I</sup> = $\{\}$                                                             |
| intersectionOf( $C_1 C_2 \dots$ )       | $C_1 \sqcap C_2$ | $(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$                                                       |
| unionOf( $C_1 C_2 \dots$ )              | $C_1 \sqcup C_2$ | $(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$                                                       |
| complementOf( $C$ )                     | $\neg C$         | $(\neg C)^I = \Delta^I \setminus C^I$                                                         |
| oneOf( $o_1 \dots$ )                    | $\{o_1, \dots\}$ | $\{o_1, \dots\}^I = \{o_1^I, \dots\}$                                                         |
| restriction( $R$ someValuesFrom( $C$ )) | $\exists R.C$    | $(\exists R.C)^I = \{x \mid \exists y. \langle x, y \rangle \in R^I \text{ and } y \in C^I\}$ |
| restriction( $R$ allValuesFrom( $C$ ))  | $\forall R.C$    | $(\forall R.C)^I = \{x \mid \forall y. \langle x, y \rangle \in R^I \rightarrow y \in C^I\}$  |
| restriction( $R$ hasValue( $o$ ))       | $R : o$          | $(R : o)^I = \{x \mid \langle x, o^I \rangle \in R^I\}$                                       |
| restriction( $R$ minCardinality( $n$ )) | $\geq n R$       | $(\geq n R)^I = \{x \mid \#(\{y. \langle x, y \rangle \in R^I\}) \geq n\}$                    |
| restriction( $R$ maxCardinality( $n$ )) | $\leq n R$       | $(\leq n R)^I = \{x \mid \#(\{y. \langle x, y \rangle \in R^I\}) \leq n\}$                    |
| restriction( $R$ someValuesFrom( $D$ )) | $\exists R.D$    | $(\exists R.D)^I = \{x \mid \exists u. \langle x, u \rangle \in R^I \text{ and } u \in D^I\}$ |

# OWL-DL Syntax(2)

| Abstract Syntax                                             | DL Syntax                                   | Semantics                                   |
|-------------------------------------------------------------|---------------------------------------------|---------------------------------------------|
| Class( <i>A</i> partial $C_1 \dots C_n$ )                   | $A \sqsubseteq C_1 \sqcap \dots \sqcap C_n$ | $A^I \subseteq C_1^I \cap \dots \cap C_n^I$ |
| Class( <i>A</i> complete $C_1 \dots C_n$ )                  | $A = C_1 \sqcap \dots \sqcap C_n$           | $A^I = C_1^I \cap \dots \cap C_n^I$         |
| EnumeratedClass( <i>A</i> $o_1 \dots o_n$ )                 | $A = \{o_1, \dots, o_n\}$                   | $A^I = \{o_1^I, \dots, o_n^I\}$             |
| SubClassOf( $C_1 C_2$ )                                     | $C_1 \sqsubseteq C_2$                       | $C_1^I \subseteq C_2^I$                     |
| EquivalentClasses( $C_1 \dots C_n$ )                        | $C_1 = \dots = C_n$                         | $C_1^I = \dots = C_n^I$                     |
| DisjointClasses( $C_1 \dots C_n$ )                          | $C_i \sqcap C_j = \perp, i \neq j$          | $C_i^I \cap C_j^I = \emptyset, i \neq j$    |
| Datatype( <i>D</i> )                                        |                                             | $D^I \subseteq \Delta_D^I$                  |
| DatatypeProperty( <i>U</i> super( $U_1$ )...super( $U_n$ )) | $U \sqsubseteq U_i$                         | $U^I \subseteq U_i^I$                       |
| domain( $C_1$ ) ...domain( $C_m$ )                          | $\geq 1 U \sqsubseteq C_i$                  | $U^I \subseteq C_i^I \times \Delta_D^I$     |
| range( $D_1$ ) ...range( $D_l$ )                            | $\top \sqsubseteq \forall U.D_i$            | $U^I \subseteq \Delta^I \times D_i^I$       |
| [Functional])                                               | $\top \sqsubseteq \leq 1 U$                 | $U^I$ is functional                         |
| SubPropertyOf( $U_1 U_2$ )                                  | $U_1 \sqsubseteq U_2$                       | $U_1^I \subseteq U_2^I$                     |